Santa Clara University
Scholar Commons

Interdisciplinary Design Senior Theses | Engineering Senior Theses

6-15-2017

RF Location Tracking: A Modular Antenna System Implementation

Alan Tao Feng
Santa Clara University, tfeng@scu.edu

Razma Mogharrab
Santa Clara University, rmogharrab@scu.edu

Carlos Rivera
Santa Clara University, crivera@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/idp_senior

 Part of the Computer Engineering Commons, and the Electrical and Computer Engineering Commons

Recommended Citation

Feng, Alan Tao; Mogharrab, Razma; and Rivera, Carlos, "RF Location Tracking: A Modular Antenna System Implementation" (2017). *Interdisciplinary Design Senior Theses*. 30.
https://scholarcommons.scu.edu/idp_senior/30

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Interdisciplinary Design Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY

Department of Computer ~~Science and~~ Engineering
Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Alan Tao Feng, Carlos Rivera, Razma Mogharrab

ENTITLED

RF Location Tracking
A Modular Antenna System Implementation

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE AND ENGINEERING
ELECTRICAL ENGINEERING

Ramesh Abhari 6/7/2017
Dr. Ramesh Abhari – Electrical Engineering Faculty Advisor date

6/8/2017
Dr. Ahmed Amer – Computer Science and Engineering Faculty Advisor date

Shoba Krishnan 6/7/2017
Dr. Shoba Krishnan – Electrical Engineering Department Chair date

N. Ling 6/8/2017
Dr. Nam Ling – Computer Science and Engineering Department Chair date

Santa Clara University
School of Engineering

RF Location Tracking

A Modular Antenna System Implementation

Senior Design Project Report

Students

Alan Tao Feng
Electrical Engineering
tfeng@scu.edu

Razma Mogharrab
Electrical Engineering
rmogharrab@scu.edu

Carlos Rivera
Electrical & Computer Engineering
crivera@scu.edu

Advisors

Ramesh Abhari Ph.D
Electrical Engineering
rabhari@scu.edu

Ahmed Amer Ph.D
Computer Engineering
aamer@scu.edu

June 15, 2017

Abstract

From the Amazon Prime Air drone delivery service to the usage of unmanned aerial vehicles (UAV) in military operations, recent years have seen the development of autonomous flight technologies becoming one of the major research topics in the drone industry. Tracking the geographic position of drones is a crucial part of any autonomous flight, but the common methods of drone location tracking either have too large of an error margin or require extensive environmental setup. The aforementioned issues are major roadblocks in the advancement of autonomous flight operations. The proposed solution is a new and improved method to track the location of a drone relative to a single reference point. This method will not require any environmental setup and offers a greater degree of precision than the commonly used Global Positioning System (GPS). The designed proof of concept model, which is a completely modular and self-reliant radio-frequency (RF) based location tracking system, was built to show the viability of this new drone tracking method. The tracking system can determine the relative location of a radio-frequency source with only one receiver module. By requiring only one receiver, this tracking system eliminates the need to set up a triangulation zone. Additionally, optimizing the tracking system to generate a location from the RF telemetry signals needed in user-drone communication, the solution effectively presents an efficient manner to track a drone without the need for additional attachments. The proposed solution introduces a novel method that has the potential to vastly improve autonomous flight development and push it to full realization and fruition.

Acknowledgements

The team would like to thank the following people for aiding the development of this project throughout the academic year:

- Dr. Ramesh Abhari
- Dr. Ahmed Amer
- Victor Figueroa
- Nicholas Mikstas

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement

Tracking the location of drones is an essential part of all autonomous flight operations. To ensure safety, the location of a drone must be tracked with high precision during autonomous flights. The commonly used Global Positioning System simply cannot offer the precision needed to pinpoint the location of a commercial drone. For location tracking systems that are precise, they require placing a plethora of sensors or WiFi modules in the area of operation to create a triangulation zone. There are many autonomous drone operations where the users do not have the luxury to perform any environmental setups. Drone deployment for military operations, search and rescue, and personal use in remote areas are just some of the situations where a triangulation zone may not be available.

1.2 Current Location Tracking Methods

Current location tracking methods include the use of Global Positioning System (GPS), WiFi, and image/video processing to determine an object of interest's position. As mentioned beforehand, shortcomings in these particular location tracking methods stem from either inaccuracy or need from extensive environmental setup that either limit use or prove to be too large an inconvenience.

1.2.1 Global Positioning System (GPS)

GPS is currently one of the most common methods of drone tracking. It uses at least 4 satellites (3 being the theoretical minimum) for triangulation. The usefulness of GPS comes from the fact that it can determine the location of its user in most places around the globe without needing Internet or telephone reception. However, even the best GPS hardware technology today only have a maximum accuracy of about 4 meters RMS (root mean square) in open areas. Additionally, he accuracy of GPS decreases significantly more in areas with environmental obstructions. In many indoor areas, GPS simply does

not work. Considering the fact that the diagonal size of most commercial professional-grade drones is about 350 millimeters, GPS simply does not offer the precision and reliability to pinpoint the location of most modern-day drones.

1.2.2 WiFi

A WiFi based location tracking system offers much better accuracy than GPS. However, in order to use WiFi for location tracking, a WiFi triangulation zone must be present. A WiFi triangulation zone is created by having multiple Wireless Access Points (WAP) placed in the intended area of operation. This setup is needed because to determine the arrival angle for an incoming signal, multiple WiFi receivers must serve as reference points and process the distance of the incoming signal source to triangulate its location. Considering the environmental drawbacks in this method, WiFi, though more accurate than GPS, proves to be a costly tracking solution that takes both numerous WAPs and extensive amounts of time to properly setup and calibrate.

1.2.3 Image/Video Target Localization

A somewhat overlooked, but still useful, method of tracking the location of drones is through the use of cameras. By using image-processing techniques on the video feed received from on-board or off-board cameras, one may use software in order to track a drone. Naturally, this method comes with the most unreliability. The solution only works when there is a line-of-sight connection between the camera lens and the drone. A major drawback of this system is that even a novice drone pilot can break the line-of-sight connection between the drone and the camera, thus rendering this method useless. Additionally, there remains the problem of requiring a preemptive setup similar to that of the WiFi zones. Lastly, video feeds for location tracking require a clear target to focus on. This means that the drones themselves need a clear visual marker. Furthermore, said visual marker must continually be facing the camera being used. This drastically limits the movement of the drone and results in an additional liability. Should there ever be an irrelevant object in the environment with such a visual pattern/cue, the tracking system might register it as an object of interest and focus on its position.

1.3 Proposed Solution

The proposed solution to the location tracking issue is a system that can determine the relative location of an incoming RF source using only one RF receiver while offering a degree of precision greater than that of GPS. This system will eliminate the need for environmental setup prior to drone autonomous operations, making it completely mobile. Users intent on drone tracking simply need place the receiver anywhere they desire; the system will then be able to track the location of a drone that is within a set range from the receiver. The ultimate goal is to test the viability of a possibly innovative method in drone location tracking. Pushing towards proving the theory is the primary goal of the solution and the additional components in the implementation serve to supplement this core functionality and push the external use of the location data produced by the tracking system.

1.4 Project Objectives

The goals of this project are highlighted through a number of key objectives. Those that define the hardware's goal are as follows

- Function with one RF receiver
- Detect an RF source within a set range
- Locate the RF source (relative distancing and direction)
- Operate at 915 MHz frequency (drone telemetry frequency)

The software, more specifically the Raspberry Pi application, intends on accomplishing the following.

- Process digital signals and extrapolate a location
- Display RF source's position on a graphical user interface (GUI)
- Display information with a reasonable refresh rate

2 Design and Implementation

To locate an RF source, the tracking system must determine the relative direction and distance of the incoming signal. This can be done using properties of spiral and dipole antennas, respectively. The following is an overview of the location tracking method used in this project.

2.1 Theoretical Background

The positioning description of an RF source is essentially divided up into two parts: the angle of the incoming RF source and the distance of the RF source. By using these two pieces of information, a location can be generated. However, a limitation in this approach is that a location can solely be mapped for a singular two dimensional plane rather than a three dimensional space. For this version of the system, such capabilities are both necessary and sufficient as the use cases are encompassed by the assumption of singular planar movement.

2.1.1 Direction

The system uses spiral antennas to determine the direction of an incoming signal. Three separate bi-directional spiral antennas are placed 120° apart. The received power of a spiral antenna is a function of an RF signal's incoming direction in the theta plane. Using this directionality property, the angle of an incoming RF signal is determined by comparing the difference in received power between any two pairings of the three spiral antenna receiver module [1]. These comparisons not only allow one to extract the angle of the RF source relative to a pairing of spiral antennas, but also aid in determining which sector the RF source is in.

The aforementioned sectors are defined by the three antennas that form the receiver. As shown in Figure 2.1, each sector is defined as a plane encompassed by three 120°spectra whose start and end points are defined by the spiral antennas.

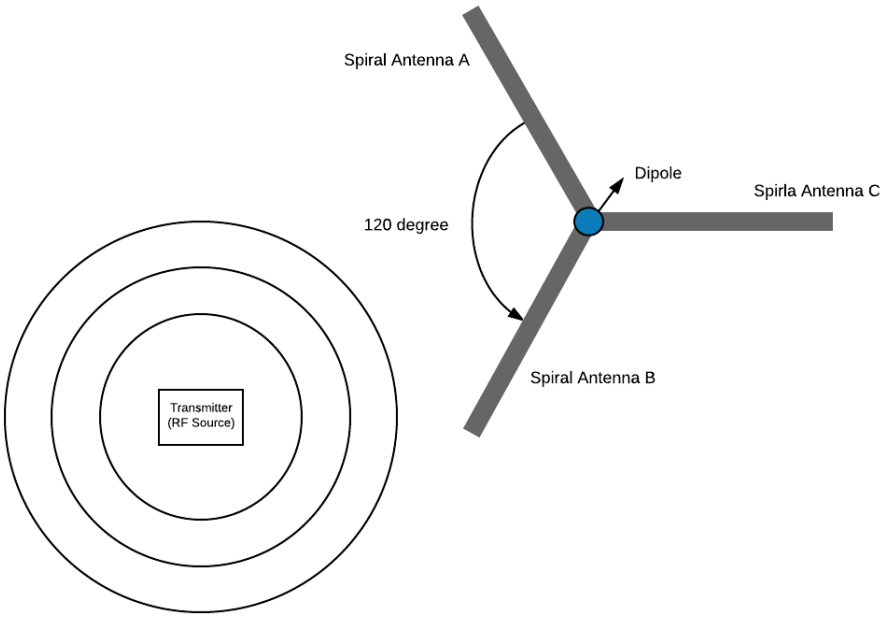


Figure 2.1: Top-down view of the location tracker.

2.1.2 Ranging

The distance between an RF source and the location tracker is determined using a dipole antenna. An omni-directional dipole antenna is placed in the center between the three spiral antennas. The received power of a dipole antenna does not depend on an RF signal's incoming direction in the theta plane. The only way to change the received power of a dipole antenna is to change the distance between the RF source and the antenna. Taking advantage of the observed omni-directionality, the distance between an RF source and the location tracker is determined by the received power of a dipole antenna [2].

2.2 System Level Overview

Prior to implementing the solution, the location tracking system was divided into three main components: the antenna front end, the receiver module hardware, and the software application and interface. Separating the tracking system into these primary components

facilitated delegation of duties and allowed for clear component goals. The antenna front end allows the system to receive radio-frequency signals. The receiver module measures and compares received power levels for all antenna pairs and then converts those measurements to digital data. Finally, the software interface translates the receiver measurements into location information of an RF source and displays it on a GUI.

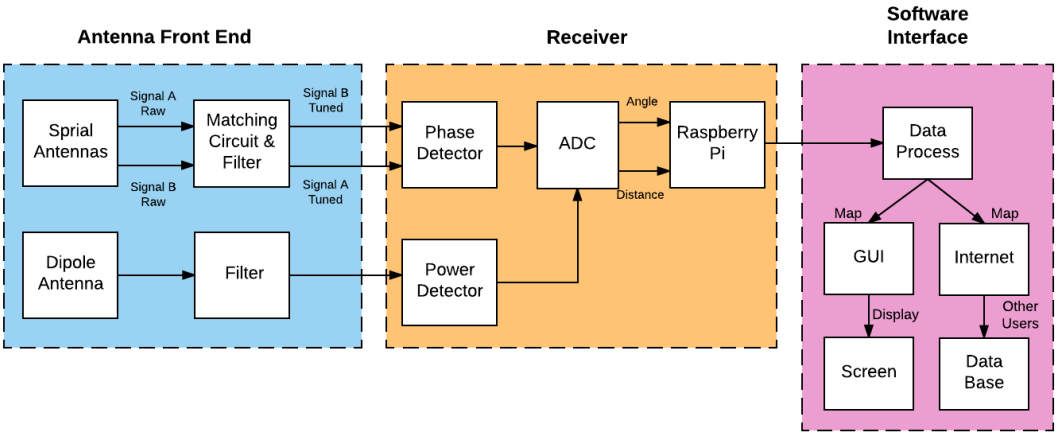


Figure 2.2: Functional block diagram.

Figure 2.2 shows the high-level architecture of the RF Location Tracking system. At the antenna front end, two types of antennas are being used. Three spiral antennas are used to determine relative direction of an incoming RF signal. A single dipole antenna is used to determine the distance between the tracker and the RF signal source. All RF signals are passed through microwave bandpass filters before being passed onto the receiver module. Phase detectors are used to compare the received power levels between any two pair of spiral antennas. An RF power detector is used to measure the received power of the dipole antenna. Both the phase detector and RF power detector output analog DC voltages, which the Raspberry Pi does not natively support. An external analog to digital converter was used to convert these analog voltages to digital signals that the Raspberry Pi could handle. Figure 2.3 shows this setup. Data from the phase detector and RF power detector were processed in order to generate the RF source's location information. Finally, a graphical user interface is used to display the location information on a screen. If an Internet connection is available, all the data from the receiver module can be uploaded to a database so other device can access the information.

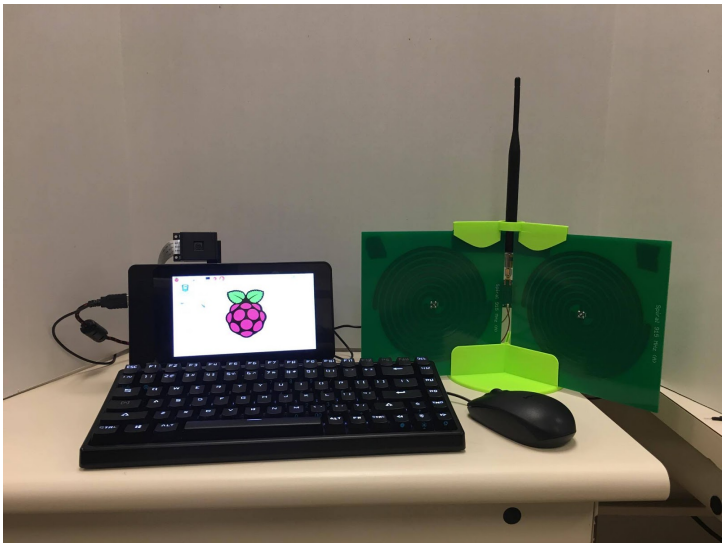


Figure 2.3: Full system setup.

2.3 Risk Analysis

In order to maximize the number of milestones achieved in the amount of time allocated, potential risks were recorded and those with the highest potential impact were more carefully handled.

Table 2.1: Risk Analysis Table

Risk	Consequence	Probability	Severity	Impact	Mitigation Strategy
Ill defined requirements	A useless product is designed	0.8	9	7.2	Build product in verified phases
Run out of time	Cut back features	0.7	8	5.6	Ensure team member scheduling
Unexpected implementation difficulty	Time to deployment increased	0.5	6	3.0	Exclude unneeded functionality
Component failure	Jeopardizing the system	0.3	6	1.8	Develop test plans for components

2.4 Requirements

Requirements for the RF Location Tracking system are broken down into functional and non-functional. Functional requirements are quantifiable characteristics of our system that make it useful. The functional requirements are as follows.

- Operate in the ultra high frequency (UHF - 915MHz) range
- Detect RF sources for distances up to 6 meters with a resolution of 5 centimeters
- Have a maximum angular error of 1 degree (10 centimeters arc length at max distance) in a 120°sector

Nonfunctional requirements are qualities that specify and define the criteria that evaluate the operation of a system. The nonfunctional requirements are as follows.

- The mapping GUI will be user friendly
- The system will be easily maintained
- The system will be easily tested
- Be a modular system
- Have a scalable nature
- Have core functionality be Internet independent
- The transmitter modules will be low cost and use off-the-shelf components

2.5 Technical Constraints

The positioning industry is currently limited to GPS and radar. These have detrimental limitations such as poor resolution, inadequate signal reception, and an overall dependence on external resources for information (WiFi, satellites, etc.). The following are a number of constraints on the system's implementation.

- Use UHF (915MHz) to avoid interference issues
  - Avoid busy frequency bands, such as 2.4GHz (WiFi and Bluetooth)

- Use a single reference point (the receiver module)
  - As the solution aims to create a location tracking system that minimizes setup, the only point of reference is the receiver module.
- Use only amplitude and phase to find location/orientation
  - Difference in amplitude and phase between all the antennas give us the location of the transmitter relative to receiver

As a follow up to the technical constraints, there is also the matter of the current technology out on the market. Currently, there are three competitors in the market that provide technology similar to that used in our system.

- Apple iBeacons
  - Uses Bluetooth Low Energy to communicate advertisement information
- Zebra
  - Uses RF to track location of warehouse items
- POZYX
  - Uses at least 5 WiFi points ("Anchors") to triangulate object of interest

The constraints follow the format of the functional requirements, but rather than placing restrictions on the problem, it restricts the solution. Our location tracking system will have to conform to the current regulation standards set forth by the FCC all while showing a degree of accuracy that, at minimum, surpasses that of GPS.

- FCC Maximum Permissible RF Exposure Regulations
- FCC Radio Spectrum Allocation
  - System must operate in the amateur radio frequency spectrum in order to avoid FCC violations.
  - Amateur radio spectrum
    - 3.6 MHz - 4.0 MHz

    - 14 MHz - 14.35 MHz
    - 50 MHz - 54 MHz
    - 902 MHz - 928 MHz
    - 2.4 GHz - 2.45 GHz
- The system's location precision and resolution must be above that of GPS.
- The total cost of the system must be below GPS.
- The antenna size must be portable.
- The system's power consumption must be below GPS modules.

3 Antenna Front End

3.1 Archimedian Spiral Antenna

Archimedean spiral antennas are used to determine an incoming RF signal's angle of arrival on a 2-dimensional (XY) plane. The left image in Figure 3.1 is the CAD drawing of an Archimedean spiral antenna done in the electromagnetic simulation software CST STUDIO SUITE. The CAD drawing was used in simulations for this project. The right image in Figure 4 is one of the physical spiral antennas used in the system. It was fabricated by a Chinese company named DreamCatcher.

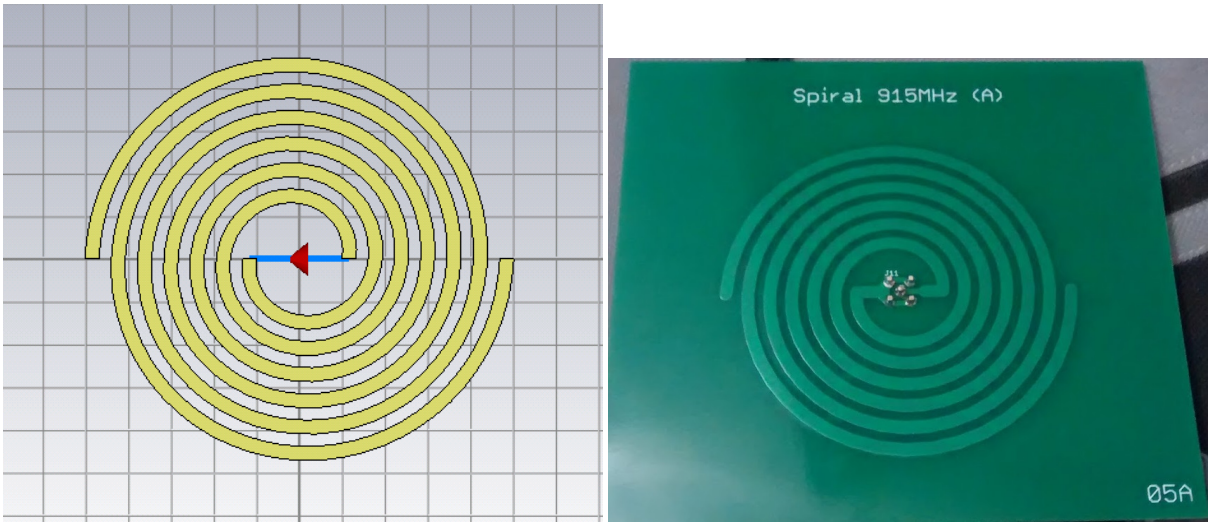


Figure 3.1: Archimedean spiral antenna.

The reasons for choosing Archimedean spiral antennas for this application are as follows.

- Received power of a spiral antenna is dependent on the angle of arrival for an incoming RF signal in the theta plane.
- Spiral antennas are bi-directional. The antenna can receive and radiate from both front and back.
- Relatively high antenna gain allows for an adequate signal reception range.
- Circular polarization. Transmitter orientation does not affect signal reception.

3.1.1 Spiral Antenna Background

The Archimedean spiral antenna is known for having a wide bandwidth. The bandwidth of a spiral antenna can be increased by adding more turns. By having two arms, spiral antennas do not require a ground plate. The signal is fed at the center and in between the two arms, one arm being the positive end the other being the negative end of the signal. The operating frequencies for an Archimedean spiral antenna are determined by the outer radius and inner radius (from center to the arms), as shown in Figure 3.2.

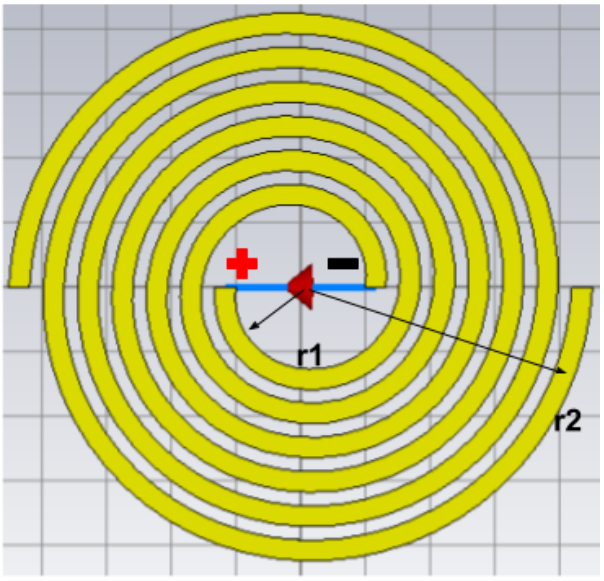


Figure 3.2: Spiral antenna with inner (r1) and outer (r2) radii.

The low frequency operating point of the spiral antenna is determined by the outer radius.

$$f_{low} = \frac{c}{2\pi r_2} \tag{1}$$

The high frequency operating point of the spiral antenna is determined by the inner radius.

$$f_{high} = \frac{c}{2\pi r_1} \tag{2}$$

3.1.2 Spiral Antenna Radiation Pattern

A single spiral antenna has the radiation pattern shown in Figure 3.3 (left). This radiation pattern is simulated in CST using the spiral antenna CAD drawing in Figure 3.1.

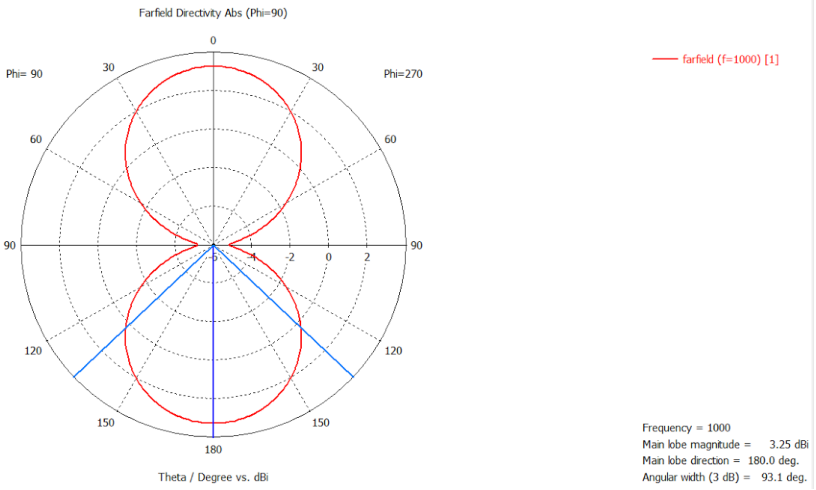


Figure 3.3: Archimedean spiral antenna radiation pattern.

Since the RF location tracking system uses three spiral antennas in total, a CAD drawing with all three spiral antennas was also made. Through CST simulation, the radiation pattern of a single antenna was found to be mostly unchanged even when two other spiral antennas were placed in close proximity. Figure 3.4 (right) shows the CAD drawing of three identical spiral antennas in CST. Figure 3.4 (left) shows the simulated radiation pattern for one of the three antennas.

The radiation pattern shows that the received power of a spiral antenna is dependent on the angle of arrival for an incoming RF signal in the theta plane. When an incoming signal is directly perpendicular to the spiral traces, or the face of the antenna, the signal will be received at maximum power level. The left side of Figure 3.5 shows the condition at which a spiral antenna will receive a signal at maximum power level. When an incoming signal is directly parallel to the spiral traces, the antenna will receive the signal at minimum power level. The right side of Figure 3.5 shows the condition at which a spiral antenna will receive a signal at minimum power level. The bi-directionality of the

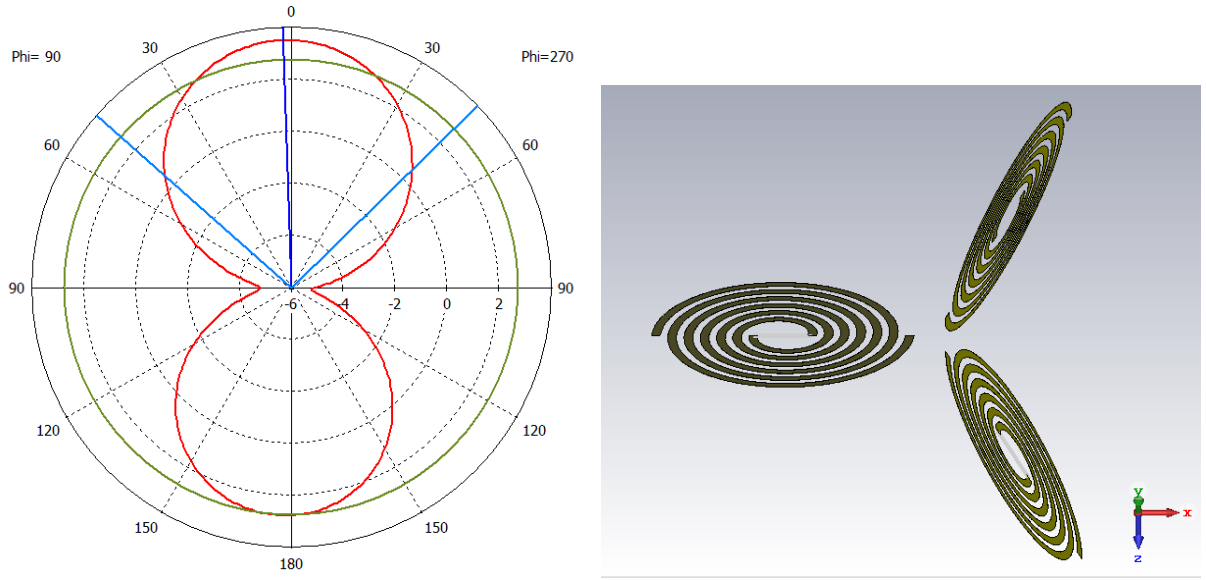


Figure 3.4: Antenna configuration and radiation pattern.

spiral antennas is shown through its radiation pattern.

3.1.3 Determining Direction in One Sector

By placing three identical spiral antennas 120° apart, space is divided up into three sectors for the tracking system. A top-down view of the system shown in Figure 3.6 describes how the three sectors are divided.

A single sector is created using two spiral antennas. Figure 3.6 shows a sector created by two identical spiral antennas, Antenna A and Antenna B. Antenna A lays directly on the XY plane and Antenna B is rotated 120° along the Y-axis

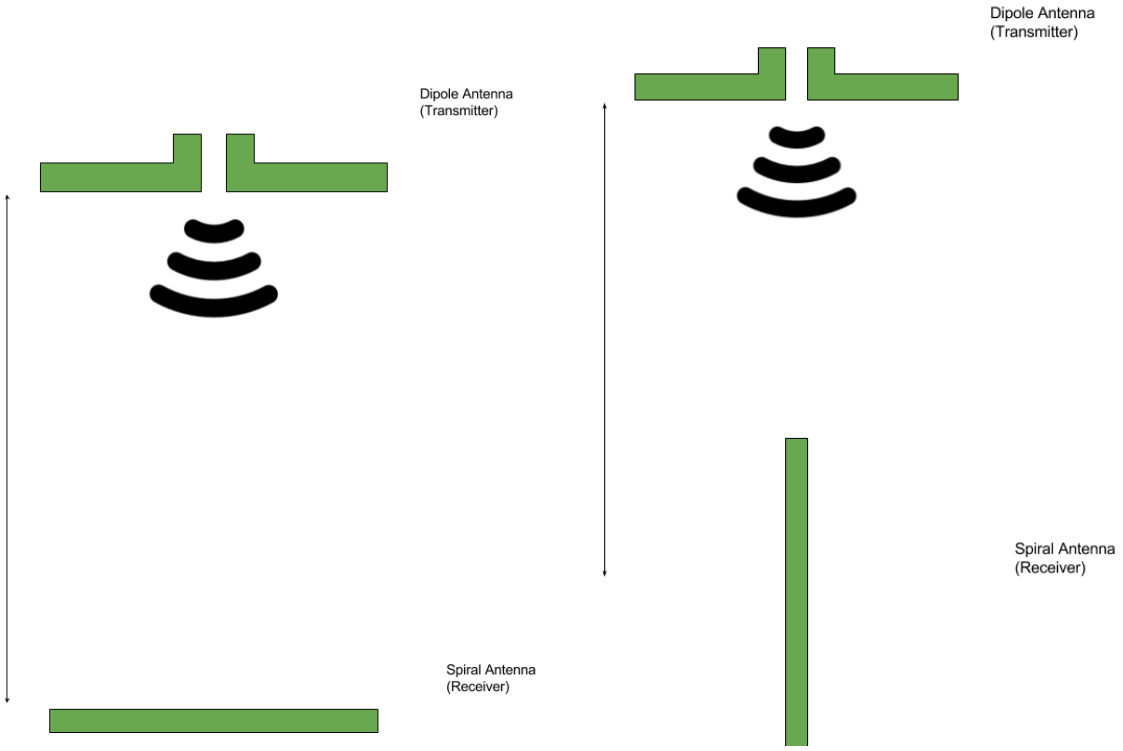


Figure 3.5: Maximum and Minimum Power Level Conditions.

The direction of an RF source that is located within this sector can be determine using the radiation pattern of the two antennas. The two figures below, Figure 3.7 and Figure 3.8, are the 3-dimensional radiation pattern of the two antennas taken from the same XY plane. Figure 3.7 and Figure 3.8 (left) are for Antenna A. Figure 3.7 and Figure 3.8 (right) are for Antenna B. The red zone shows the area where an antenna is receiving at maximum power level. The green zone shows the area where an antenna is receiving at minimum power level.

From the two radiation patterns, it is shown that if an RF signal is coming in from the right side of the sector, antenna A will be receiving that signal at close to maximum power level and antenna B will be receiving it at minimum power level. That is because the far right of the sector is located at the red zone of antenna A and green zone of antenna B. As the RF source move from the right to the left side of the sector, antenna A starts to receive less power and antenna B starts to receive more power. The far left side of the sector is located at the green zone for antenna A and red zone for antenna

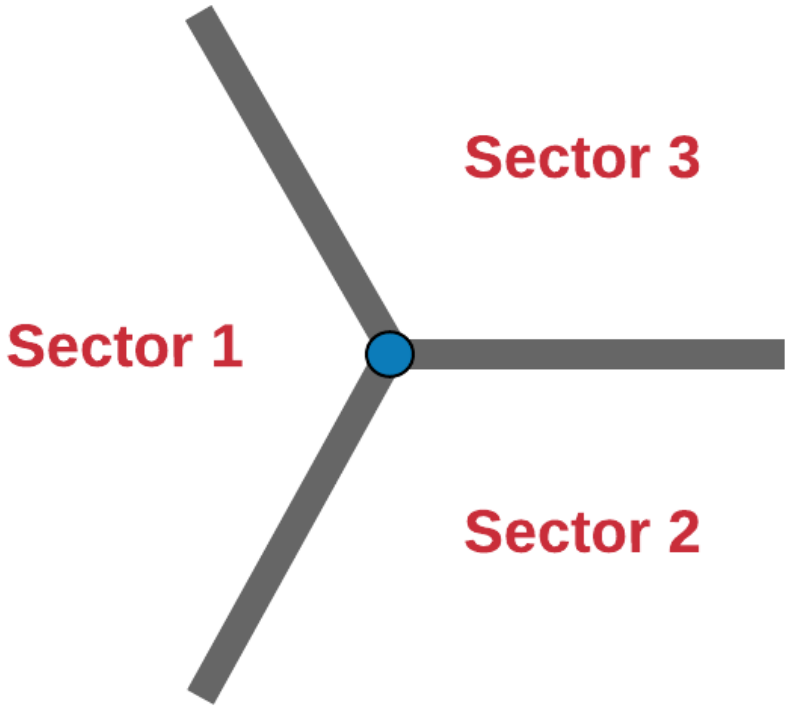


Figure 3.6: Sectors Division.

B. Thus, the ratio of received power between the two antennas changes as an RF source moves across the sector.

3.1.4 Spiral Antenna Evaluation

The operating frequency of a spiral antenna is evaluated by measuring the scattering parameter at the feed port. S11 (voltage wave reflection) for the physical antenna is measured using a vector network analyzer. [3], [4]. Figure 3.9 shows the S11 plot for the physical spiral antenna used for the system.

The S11 plot in Figure 3.9 shows that the spiral antennas used in the system are optimized to work at 1.1GHz frequency. This is not exactly the 915MHz frequency the tracking system needs in order to operate, but the antenna still works well enough at 915MHz with S11 measured to be at about -11dB at that frequency.

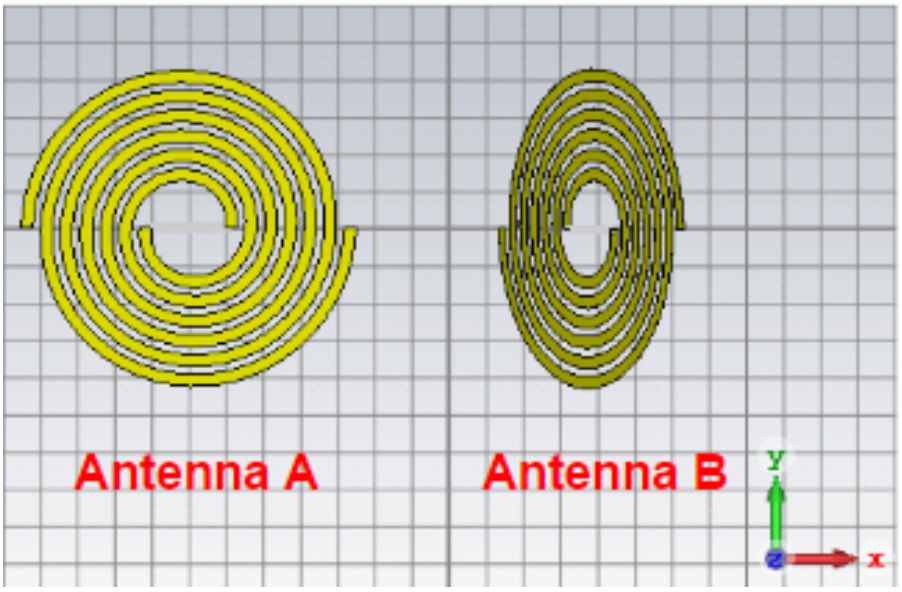


Figure 3.7: Simulated spiral antennas placed 120°apart in simulation environment.

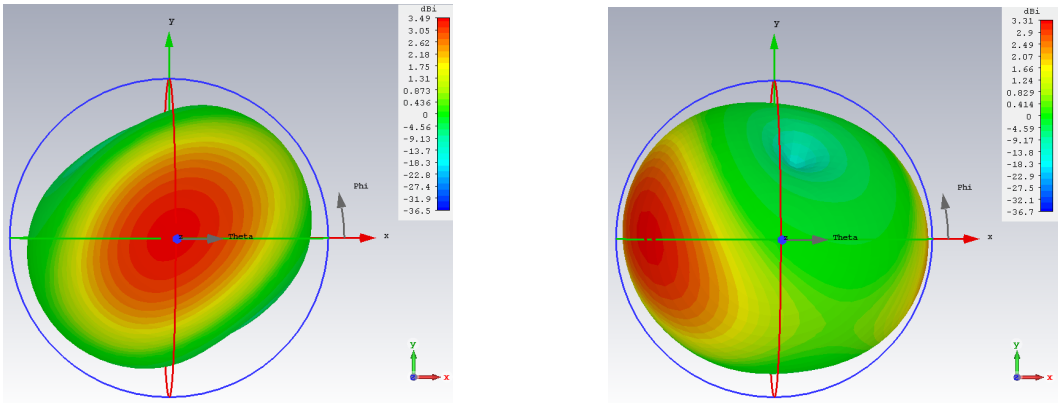


Figure 3.8: Maximum red and minimum (green) power received by simulated antennas.

3.2 Dipole Antenna

A half-wave dipole antenna is used to determine the distance between an RF source and the location tracker. Figure 3.10 shows the physical dipole antenna used for the system. It is a quad-band cellular antenna made by ADH Technology.

The reasons for choosing dipole antenna for this application:

- The received power of a dipole antenna does not depend on an RF signal's incoming

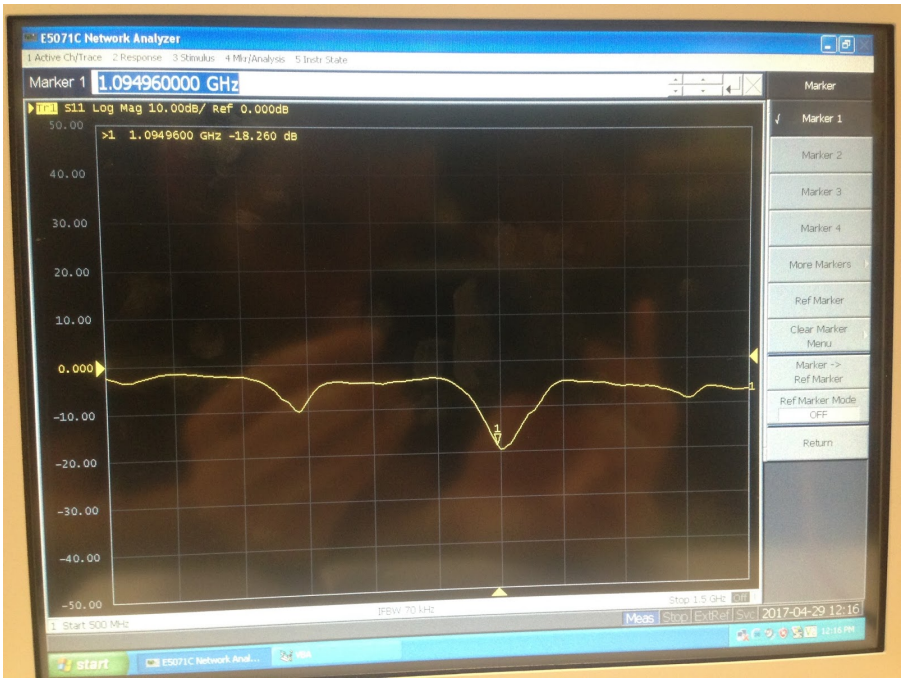


Figure 3.9: S11 scattering parameter (reflection coefficient minimum at 1.1GHz).

direction in the theta plane.

- Widely available to purchase in the market.
- Dipole antennas are used in drone telemetry transmission

Figure 3.10: Quad-band cellular dipole antenna.

3.2.1 Dipole Antenna Background

Half-wave dipoles are one of the most commonly used antennas in wireless communication. Its operating frequency is dependent on its physical length. The optimal length is half the wavelength of the operating frequency. The gain of a typical dipole antenna is 2.15dBi. [5], [6].

3.2.2 Dipole Antenna Radiation Pattern

The radiation pattern of a typical dipole antenna is shown in Figure 3.11. Figure 3.11 (left) shows the dipole radiation pattern in 3-dimensions and Figure 3.11 (right) shows the radiation pattern on the theta plane. Both of the figures were captured from CST using a sample 2.4GHz dipole antenna CAD drawing. The radiation pattern of a 915MHz dipole would have the same radiation pattern.

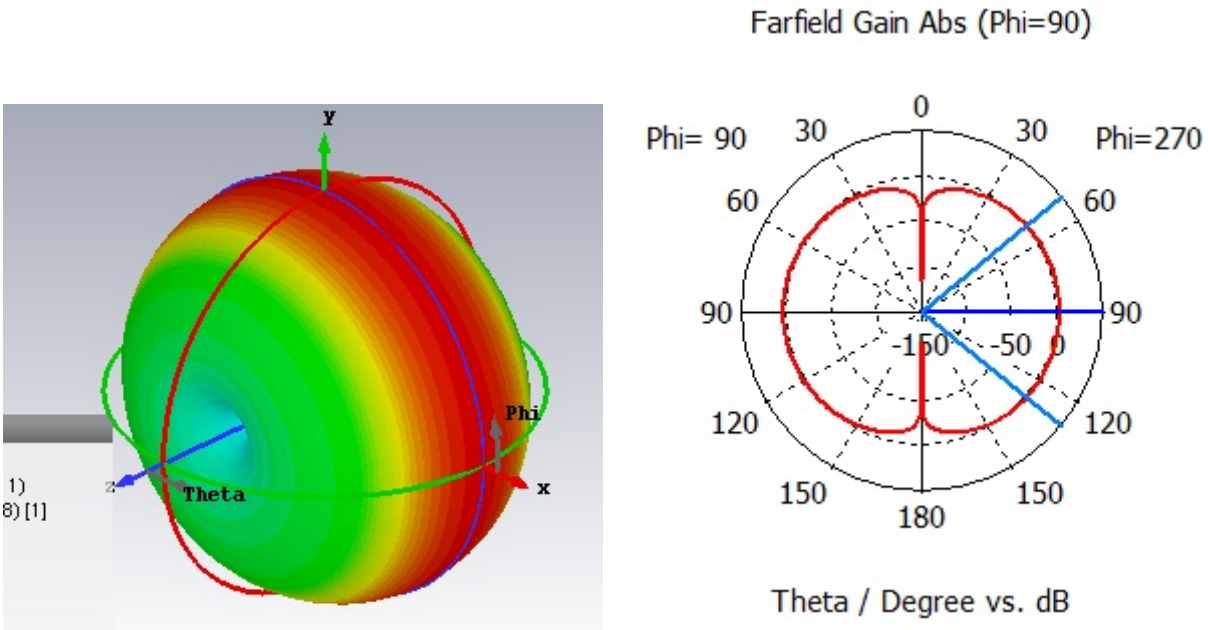


Figure 3.11: Power received as a function of direction for dipole antenna.

3.2.3 Ranging

From the radiation pattern, it is shown that the received power of a dipole antenna does not depend on the direction of an incoming signal. The only way to change the received power of a dipole antenna is to change the distance between the RF source and the antenna. Using this omni-directionality property, the distance between the RF source and the location tracker can be determined by simply measuring the received power of the dipole antenna and empirically mapping it to a physical distance. A high received power level means that the RF source is closer to the tracker. A low received power level means that the RF source is further away.

3.2.4 Dipole Antenna Evaluation

The operating frequency of a dipole antenna is evaluated by measuring the scattering parameter at the feed port. S11 for the physical antenna is measured using a vector network analyzer. Figure 3.12 below shows the S11 plot for the physical dipole antenna used for the system.

The S11 plot shows that the dipole antennas used in the system is optimized to work at four frequency ranges. The GSM band (880MHz to 960MHz) is one of the four operating frequency ranges. 915MHz falls within the operating frequency range of this dipole antenna. The S11 is measured to be about -10dB at 915MHz

3.3 Microwave Bandpass Filter

To ensure that RF signal being passed from the antennas to the receiver are only those that have the desired frequency (915MHz), microwave bandpass filters are implemented to the system. The microwave filter used in the system is the Crystek SAW Bandpass Filter with a center frequency of 915MHz. The Crystek SAW Bandpass Filter has a small bandpass width. Figure 3.13 is a picture of the bandpass filter.

Without the bandpass filter, the spiral antenna receives the 915MHz signal (transmitting at 12.5 dBm) at about -35 dBm, but the antenna is also receiving signals between 760MHz

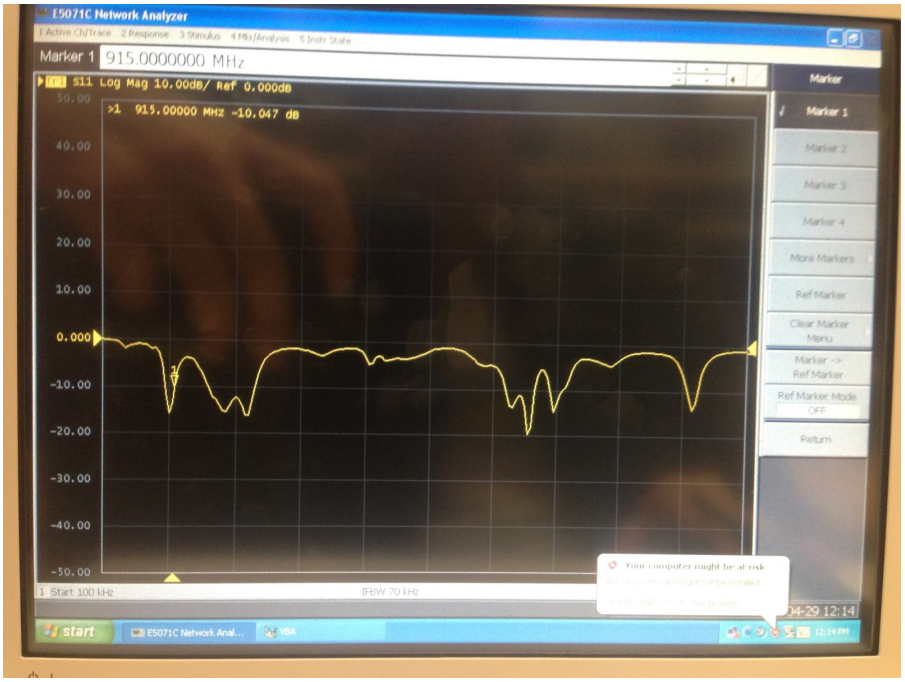


Figure 3.12: S11 scattering parameter at feed port of the dipole.

Figure 3.13: Crystek SAW 915MHz band pass filter.

to 880MHz at about -45 dBm. These signals interfere with the system.

With the bandpass filter, the same 915MHz signal (transmitting at 12.5 dBm) is being received at -39 dBm by the spiral antenna. The lower received power is due to the

insertion loss of the bandpass filter. Even though there is a loss of received power, the signals that were interfering with the system are now being filtered out. Only the 915MHz signal is being passed from the antenna to the receiver when the bandpass filter is added.

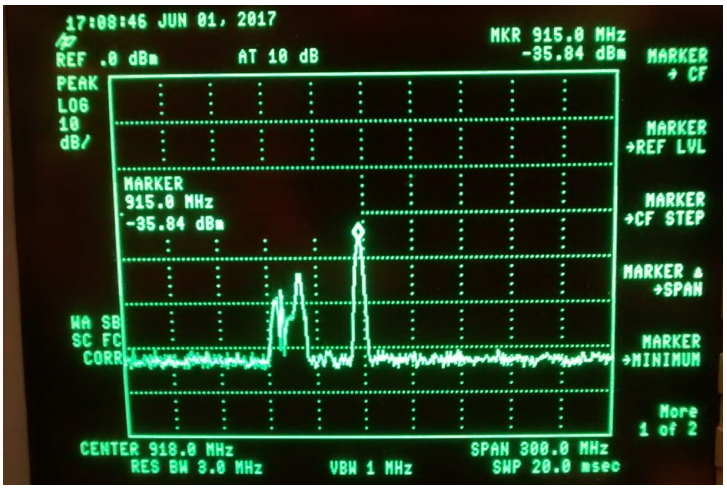


Figure 3.14: Spectrum Analyzer Reading Without Filter

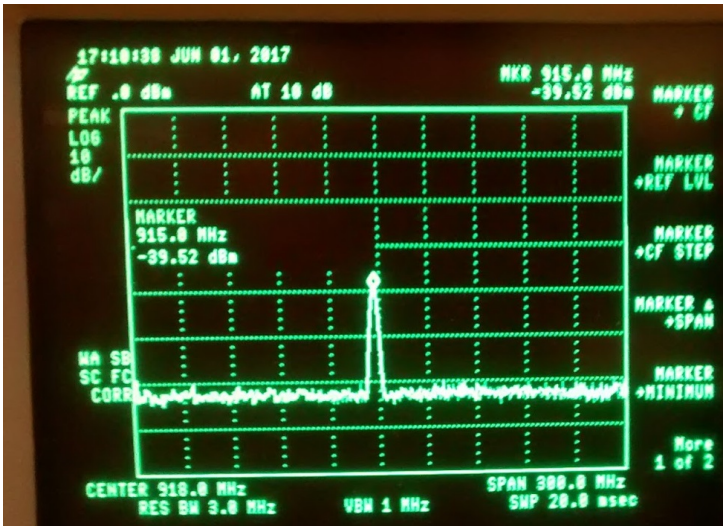


Figure 3.15: Spectrum Analyzer Reading With Filter

4 Receiver Module Hardware

The methodology for finding the relative direction of a radio frequency source is simple. The location tracker works by comparing the received power levels of the two Archimedian spiral antennas on the antenna front end. This ratio of power levels changes from low to high, or vice-versa, as the RF source moves from one end of the sector to the other end.

The module that is used to calculate this power ratio is the AD8302 phase detector board from Analog Devices (Figure 4.1). This board accepts two SMA inputs with power levels ranging from -60dB to 0dB (decibel scale), and uses those two input signals to calculate meaningful values and provide usable outputs. The module provides two DC voltage outputs, one of which corresponds to the difference in phase of the two input signals, and the other corresponds to the ratio of their magnitudes, as governed by Equation (3). The AD8302 module was chosen because it is a quality product that provides accurate gain measurement scaling at 30mV/dB, works across an impressive frequency spectrum from 30kHz to 2.7GHz, and has two useful outputs (magnitude and phase) if it became necessary to add additional capabilities to the product.

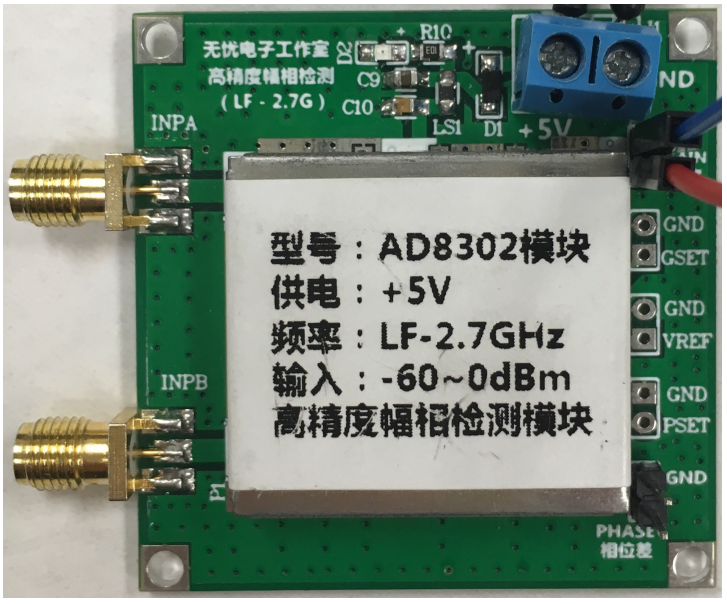


Figure 4.1: Analog Devices AD8302 phase detector.

$$20log(\frac{V_{IN,A}}{V_{IN,B}}) \tag{3}$$

For the purposes of this project, only the magnitude output of the board is being used. As shown in Appendix D, the output voltage of this module is a function of the input power ratio and ranges from 30mV to 1.8V when the RF source is spatially placed at the extremes and 0.9V when it is at the center of the sector.

While the innovation of the project lies in the clever configuration and implementation of our Archimedian spiral antennas to find the relative direction of an RF signal, the methodology for finding the relative distance took advantage of a more commonly used practice. A dipole antenna was utilized, which fed into an Analog Devices AD8317 RF Power detector (Figure 4.2). It was decided to use a dipole whip antenna for this part of the signal acquisition because it receives equal power regardless of its physical orientation. The received power only changes as a function of separation distance. This received power is then read using the AD8317 chip, processed through an analog to digital converter, and sent through a mapping algorithm to produce a distance measurement.

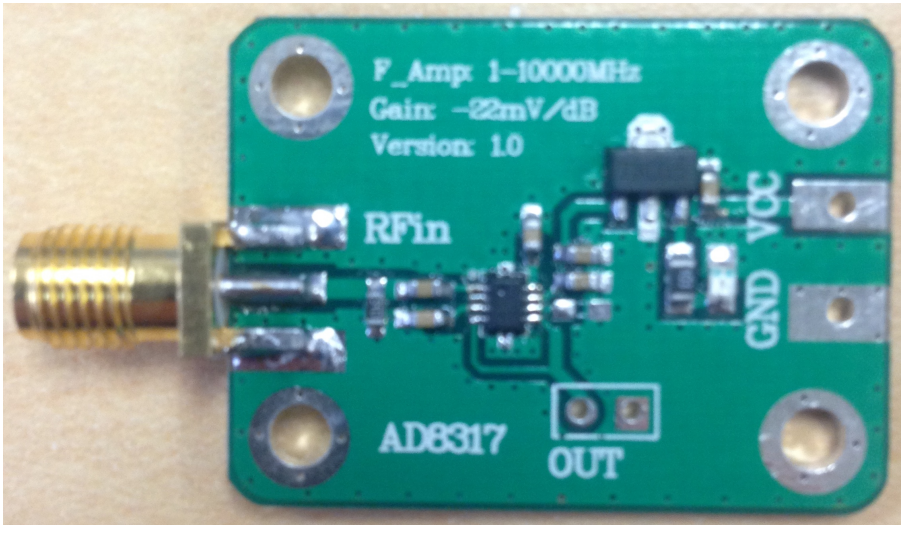


Figure 4.2: Analog Devices AD8317 power detector.

4.1 Sector Determination

To determine the relative direction of the RF source, it was imperative that an algorithm or process was created that approached the problem from a top-down perspective in order to avoid loss of generality and blind-spot error. That algorithm is as follows. First, the antenna array was split up into three sectors, which were arbitrarily name Sector A, Sector B, and Sector C. Next, the system narrows down which sector the RF source must be in. After that, the algorithm divides that sector into two parts and determines which subsector the RF source must be in. Then, when there is finally a small area to work with, the system is able to capture a much finer measurement and map it to the GUI. It was found that using this algorithm is the fastest and most efficient way to find the relative direction of an RF source given the current setup. This process is shown in greater detail in Figure 4.3.

Because the antennas are separated by 120^o instead of 90^o, it was necessary to empirically determine the new extremes of the phase detector's voltage output, which was found to be 0.55V to 1.2V instead of the 30mV to 1.8V specified in the datasheet. This was actually beneficial because, as shown in the phase detector's magnitude output graph in Appendix D, the output is very linear in that voltage range.

Having the algorithm in place, now it must be technologically implemented into the system. To begin, there was one phase detector used per sector. For example, the sector that consists of the boundary of antennas A and B (as denoted by the red stars in Figure 4.3) has one phase detector connected to the output of the two antennas, which compares their power levels. Depending on which sector the RF source lies in, the phase detector associated with that sector will produce an output voltage between our empirically-determined values of 0.55V to 1.2V. For example, if the RF source is in Sector A, the phase detector associated with Sector A will produce a an output voltage within the aforementioned range, while the other two phase detectors will produce a voltage that is not within that range.

Once it has been determined in which sector the RF source lies, the system compares the voltage of the phase detector associated with that sector to its neighbors. Depending on that ratio of voltages, it can be found whether the RF source lies within the left-half

subsector or the right-half subsector.

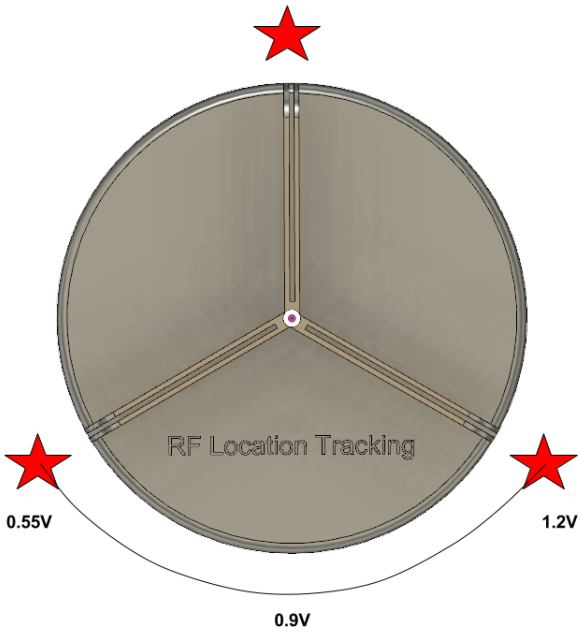


Figure 4.3: Sector determination.

Finally, after determining the subsector in which the calculations are done, the algorithm has essentially narrowed down the relative direction of the RF source to a 1/6 cut (or a 30°range) of the original circle. The final part of the algorithm is to take the voltage produced by the phase detector governing that sector and map it do a physical distance. This is, again, a process that was done empirically and the details are shown in depth in the next section.

4.2 Component Testing

With all the sub-parts chosen, in order to begin the final system testing of our product, full confidence must be had that those sub-parts all work properly. Thus, each component that we used in our project was tested thoroughly to make sure they all work as specified. To that end, the AD8302 phase detector and the ADS1015 analog to digital converter were tested, and the results are as follows:

4.2.1 Analog Devices AD8302 Phase Detector

Because the phase detector module is the main decision-maker in our project, it was tested very thoroughly by checking the magnitude output and the phase output at both the extreme values (±30dB power ratio / ±180º phase difference) and at the center value (0dB power ratio / 0º phase difference). The relative direction of an RF source is mapped onto the GUI using values that are obtained by this phase detector, so it was necessary to be confident that those values were consistent and close to the theoretical value that was expected. To show this, the outputs were tested and the results are shown in Table 2 below.

To test the phase output, we used a function generator that provided two signals that were exactly the same, but separated by a phase difference chosen by us to be 25MHz. This particular function generator was chosen because it was the only one that had the capability to provide the same signal from the same generator with a known phase difference. It was necessary to know the expected phase difference so that it could be compared to the output of the phase detector. Although the function generator's frequency output was capped at 25MHz and the project operates at 915MHz, we found that this was not a problem because 25MHz is well within the phase detector's operating range of 30kHz to 2.7GHz. This full test is shown in Figure 4.4, Figure 4.5, and Figure 4.6 for the lowest, middle, and highest outputs of the phase detector, respectively.

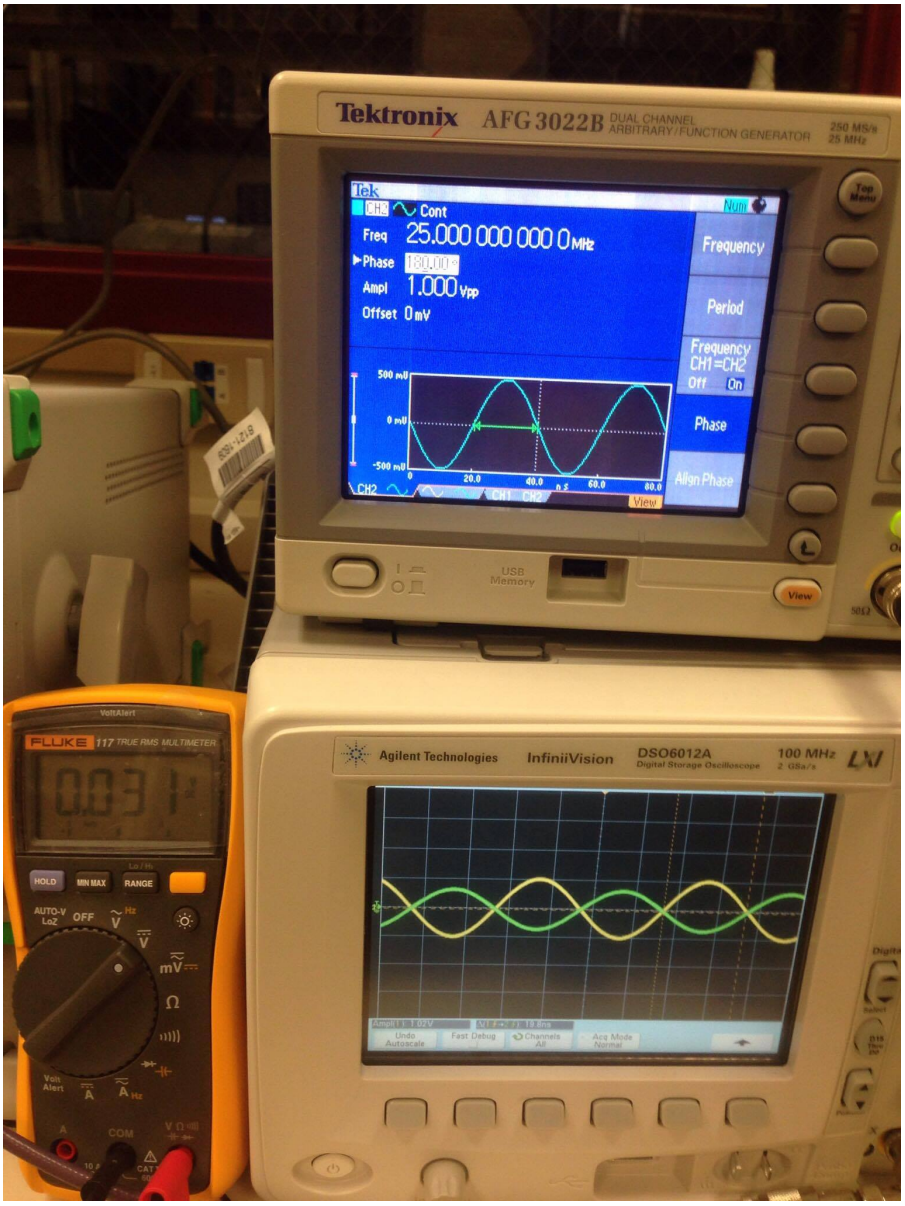


Figure 4.4: Low Output of Phase Detector (31mV vs 30mV Theoretical).

While the two measured waveforms may look different on the oscilloscope in terms of amplitude, this is simply a measurement error due to the fact that a non-ideal cable setup was required to measure both signals at the same time and one of the signals became heavily attenuated. In actuality, both signals are exactly the same, but separated by a difference in phase, so the discrepancy in the amplitudes may be justifiably ignored. The

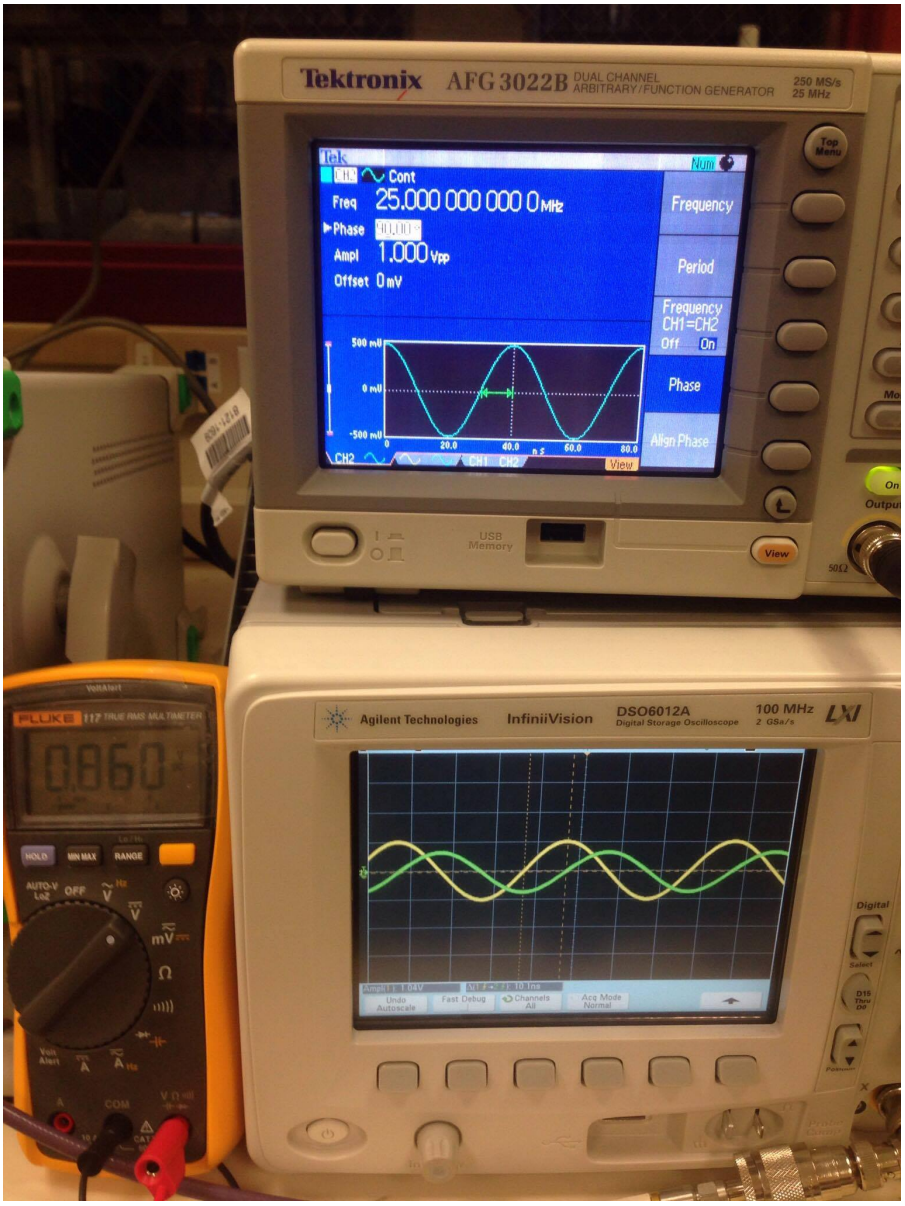


Figure 4.5: Middle Output of Phase Detector (0.860V vs 0.9V Theoretical).

team placed a voltmeter at the phase output of the phase detector and tabulated the results in Table 2 below.

To test the magnitude output, two separate function generators were used, with both at 915MHz this time, because these particular function generators were capable of providing

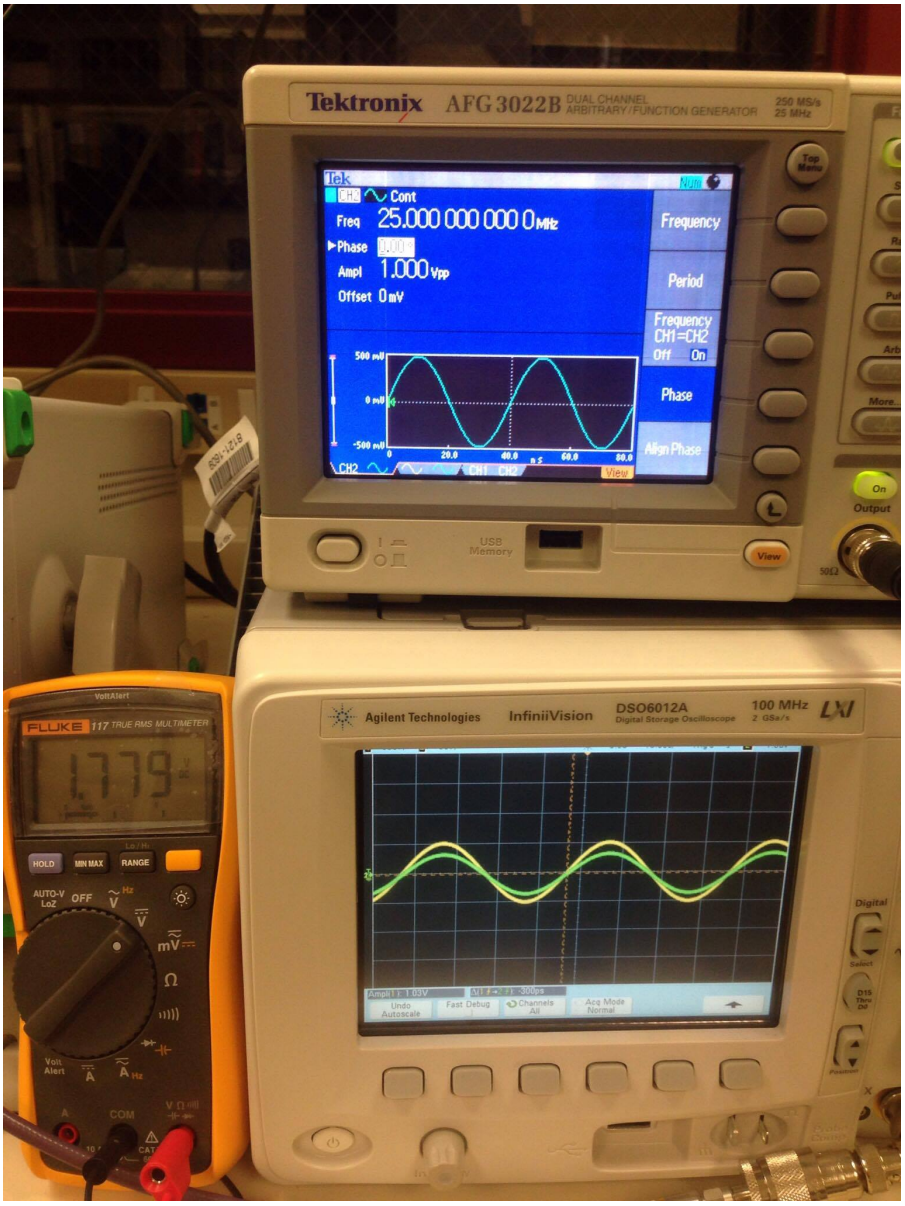


Figure 4.6: High Output of Phase Detector (1.779V vs 1.8V Theoretical).

the required frequency output. A simple setup was used with two cables of the same length and from the same manufacturer to connect both function generators to the phase detector. To calculate the theoretical voltage at the magnitude output of the phase detector, we made the function generator output the signal at power levels that were chosen by us. The ratio of those power levels was then taken and compared to the

theoretical DC voltage output determined by Equation (3) above. This was the value that was compared to the voltmeter reading on the phase detector and the results are tabulated in Table 2 below. It is important to note that all but one of the errors are less than 1%.

Table 4.1: Theoretical and Experimental Value Comparison.

Amplitude Ratio	**DC Output (Theoretical)**	**DC Output (Experimental)**	**Percent Error**
20 *log(Va/Vb)* = -30 dB	30 mV	30.2 mV	0.67%
Va = Vb	900 mV	904 mV	0.44%
20 *log(Va/Vb)* = +30 dB	1.8 mV	1.805 mV	0.28%
Phase Difference	**DC Output (Theoretical)**	**DC Output (Experimental)**	**Percent Error**
0°	30 mV	30.1 mV	0.33%
90°	900 mV	860 mV	-0.44%
180°	1.8 V	1.779 mV	-1.17%

4.2.2 Texas Instruments ADS1015 ADC

The team found it important to test the ADS1015 ADC because the first one that was purchased did not work and it was preferred to begin system testing fully confident that all of the subsystems work properly so that less time was wasted searching for things that went wrong. To test the ADC, the proper outputs (SCL, SDA, ADDR, and a receiving channel) were plugged into the Raspberry Pi's corresponding GPIO pins and an oscilloscope was connected to the SCL and SDA pins in parallel. The results were as shown in Figure 4.7, which shows the ADC clock in blue and the ADC data line in yellow. As shown, the clock edges appear to be very smooth square pulses. The clock edges attempt to ask the ADC for a response. The first seven pulses of the clock are the address, the eighth pulse is read/write, and the ninth pulse is acknowledgement. As the figure shows, everything was working properly for the ADC, so it was used in the full system testing.

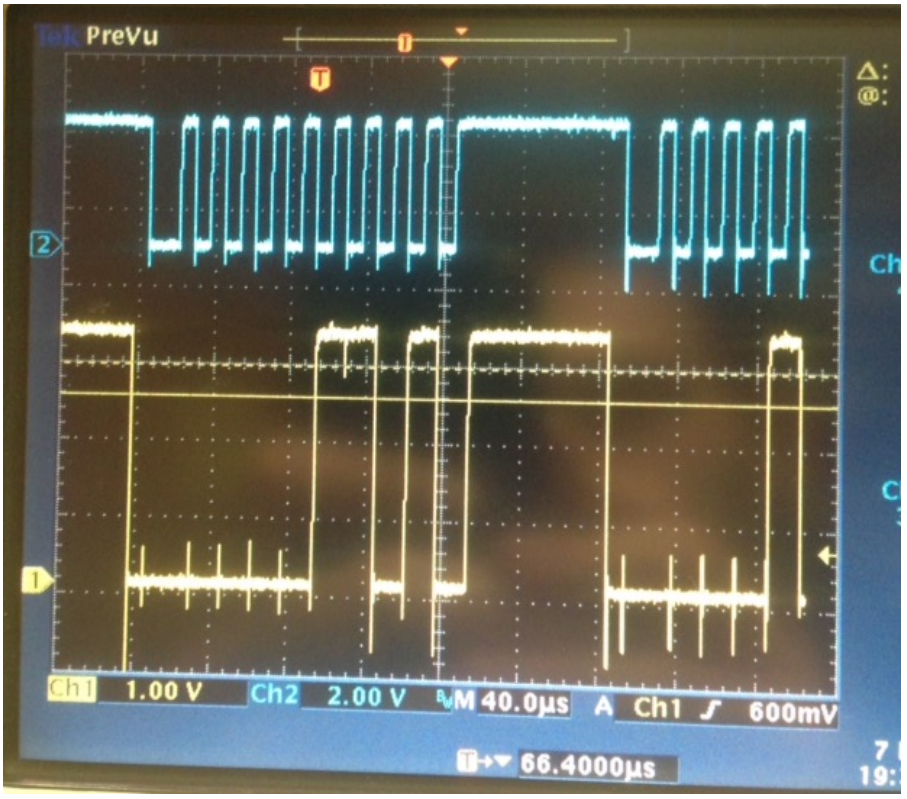


Figure 4.7: ADC Oscilloscope Test.

4.3 ADS 1015 vs ADS 1115

Two common analog to digital converters that are compatible with the Raspberry Pi 3 are the ADS1015 and ADS1115. These two ADCs provide similar capabilities, but differ in terms of number of bits and speed.

Table 4.2: Analog to Digital Converter Differences.

ADS1015	ADS1115
12 Bit Resolution	16 Bit Resolution
128 to 3300 samples per second	8 to 860 samples per second

In the final design, the ADS1015 was ultimately used due to it being not only readily available, but also because for this particular iteration of the solution it was deemed that resolution could be sacrificed for improved samples rates. This was primarily based on

the fact that the solution was to be implemented with as few bottlenecks as possible when it came to data being received. Additionally, considering that for this version of the solution the intended operating range would be decreased, the need for additional resolution would solely and simply allow for finer precision in the readings, but also, as a result display noise more prevalently. Because both of the two ADCs accounted for both positive and negative voltages, in either case the full resolution was cut in half.

Below are the specification of the ADC used in the final proof of concept model.

- ADC: ADS1015
- Resolution: 12-bit (-2048 to 2047)
- Detectable Voltage Range: $\pm$ 2.048 V

4.4 Communication Protocol

For the communication protocol between the ADC and the Raspberry Pi tablet, it was decided that it was best to use the I2C communication protocol. We made this decision because I2C makes it easy to scale up the system in that the same 2 buses (SDA and SCL) can be used to communicate between the master device (Raspberry Pi tablet) and slaves (ADCs).

5 Software Interface

Due to the intention of having the application run on the 7 inch resistive touchscreen display attached to the Raspberry Pi, the application's layout was optimized to run for said hardware. Additionally, in order to keep operations as independent as possible, the software application has had its primary features implemented on separate threads.

5.1 Overview

5.1.1 Use Cases

As the system is currently in its proof of concept phase, it can be said that there isn't an explicit customer in mind. The following elaborate on the actions that the person interacting with the system, more specifically the Raspberry Pi software application, can do.

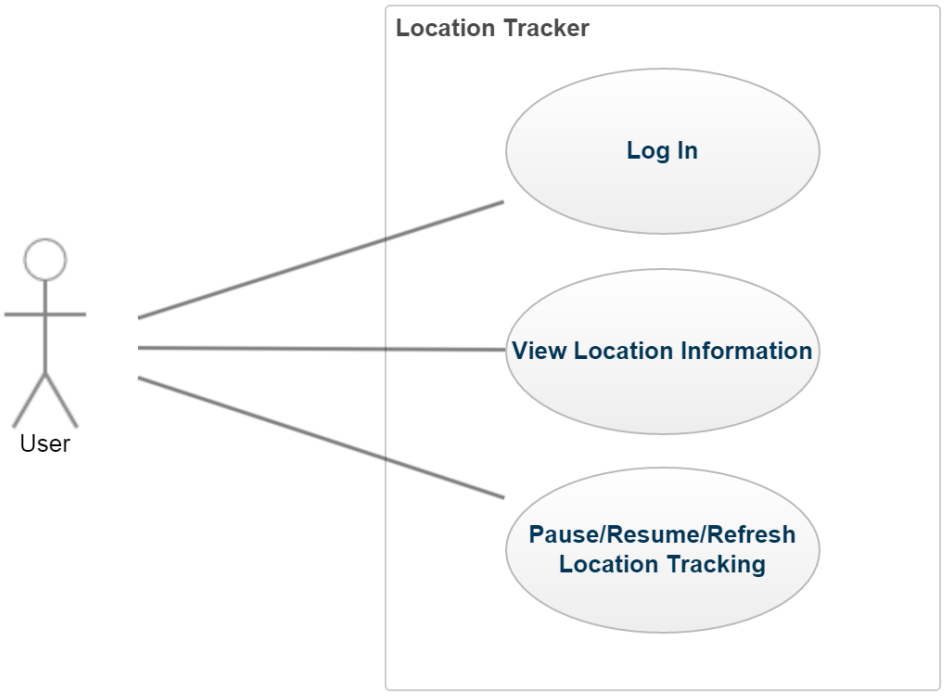


Figure 5.1: Use Case Diagram.

Table 5.1: Log In Use Case

Actor	User
Goal	Log In
Preconditions	Have RF Tracker application installed Have an Internet connection Have an account
Postconditions	User's account is checked
Steps	1. Launch RF Tracker application 2. Enter email and password 3. Press 'Log In'
Exceptions	User is not connected to the Internet

Table 5.2: Location Information Use Case

Actor	User
Goal	View location information
Preconditions	Have RF Tracker application installed
Postconditions	User observes nearby RF sources
Steps	1. Launch RF Tracker application 2. If there is an Internet connection, log in 3. View RF sources on polar plot
Exceptions	None

Table 5.3: User/Plot Interaction Use Case

Actor	User
Goal	Pause/Resume/Refresh tracker plot
Preconditions	Have RF Tracker application installed
Postconditions	User changes plot behavior
Steps	1. Launch RF Tracker application 2. If there is an Internet connection, log in 3. Press corresponding tool bar icon
Exceptions	None

5.1.2 Activity Diagram

The activity diagram is meant to highlight the processes and natural flow and progression taken in task execution. Since the primary objective of this particular iteration of the solution is to test the viability of this tracking method, user interaction was limited to

only those deemed necessary. Processing power was focused on calculations and plotting in order to maximize accuracy and read rate.

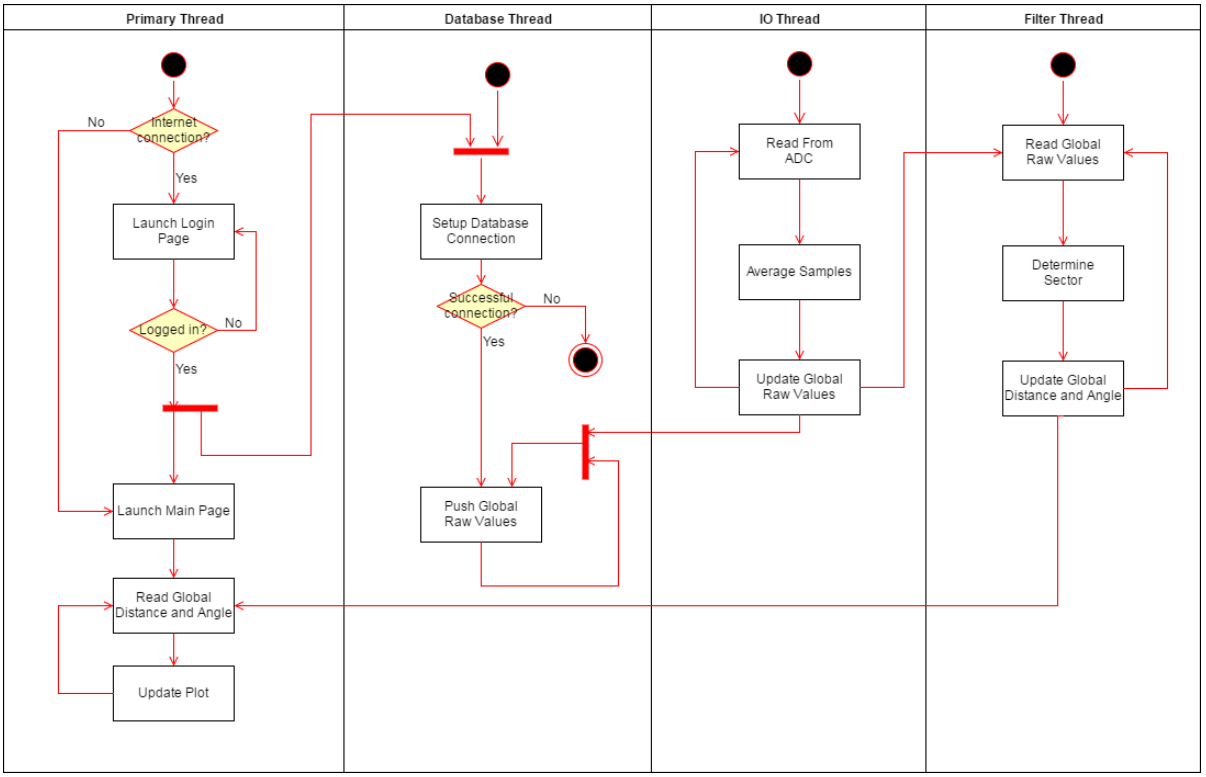


Figure 5.2: Activity Diagram.

As mentioned previously, the software application makes use of multithreading and as is indicated by Figure 5.3, there are four threads in the software application. The user primarily interacts with the primary thread as it is the one that handles the login, the user's primary interaction with the application. The remaining threads work in tandem in order to update the global variables that hold the raw and actual location values. Essentially, the IO thread interacts with the ADCs and records the numeric values for the digital signals onto the raw global variables. The filter thread then makes sense of the raw numbers and generates actual location description values. This thread, though indicated as the filter thread, does not perform any explicit and dedicated filtering. This thread has been indicated as such because it has been deemed as the optimal place in the application to insert a software filter. The database thread pushes the use of the data further by periodically reading the global raw values and uploading them to a database.

5.1.3 Architectural Diagram

The architectural diagram serves as a graphical representation of the primary high-level entities in the software application and the tasks they perform.

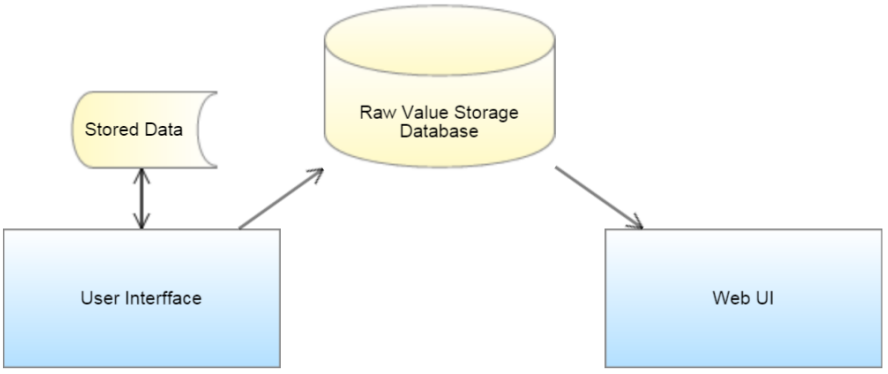


Figure 5.3: High-level architectural diagram.

At a high level, the entirety of the software is encompassed through four entities. The Raspberry Pi application encompasses two of these entities: the user interface and the saved data. Only the current data, both the raw values and location information, are stored on the Raspberry Pi, or in a more general sense, the hardware running the application. The primary user interface is written entirely in Python and made using the PyQt5 framework. The data saved on the hardware is done solely through global variables and is not saved in a file format. The database for the raw value storage is done through Firebase and the interfacing between the Raspberry Pi application and the database is facilitated through Firebase. The web user interface uses HTML and CSS for the layout and all requests and actions are written in Javascript. The layout of the web application was taken off of Bootstrap templates. Due to Firebase's natural integration with web applications, read requests of the database were done directly through Firebase's provided API and did not require an external software library.

5.1.4 Software Technologies Used

On the hardware aspect of the tracker, there were a number of commonplace hardware components used. The software was implemented and composed of commonly used tools

and frameworks. The following are the key software technologies that were used in the implementation of the application.

- Python programming language
- PyQt5 GUI framework
- Firebase
- CanvasJS

Pyrebase and Adafruit's ADS1X15 software libraries were used to interact with database and hardware respectively. Additionally, MATLAB was used during testing and data collection, but was not used in the actual software application.

5.2 Design Rationale

In defining the solution, a number of critical decisions were made regarding the approach and technologies used. The following flowchart in Figure 5.3 highlighted decisions were those deemed most important regarding the project. The complementary analysis of the potential results yielded from different possible implementations is meant to justify the solution's actual implementation

5.2.1 Implementation Programming Language

This particular iteration of the software aspect of the location tracking system was implemented entirely in Python. The reasoning behind this decision is primarily due to the ease of use of the Python programming language and the number of power tools readily available for Python applications. Additionally, considering that the solution would be implemented for and soft-released on a Raspberry Pi, early stage research revealed a large number of external support and a plethora of resources for Python-based Raspberry Pi applications. By choosing a popular language for the given hardware, possible implementation roadblocks could more likely be easily identified and resolved.

In terms of the limitations in implementing the software application with such a high

level programming language, it was found that explicit control over specific hardware functions was impossible. This limitation, primarily with thread priority and scheduling, introduced roadblocks that limited the extent to which the tasks being performed concurrently by the application could be run without conflicts or operating system-level error.

5.2.2 Software Application Framework

There are a number of popular frameworks for creating the graphical user interface aspect of a standalone application. Given that the chosen implementation language was Python, the compatible frameworks were narrowed down to Tkinter and PyQt5, a Python binding for the cross-platform C++ GUI framework. Further research indicated that Tkinter was a lightweight solution best fitted for simple applications whereas Qt was a more prevalent framework in the industry. In order to adhere to such standard and increase the solution's adjacency to that which is common practice while retaining accessibility to the tools for Python, PyQt5 was chosen as the ideal framework.

One drawback from using this particular framework was the fact that the documentation for the C++ version of Qt was more abundant than that for the Python version. Fortunately, the C++ code was easily translatable to Python code due to the classes and methods in PyQt5 being consistent and easily accessible via the class interfaces.

5.2.3 Database Platform

As there are a number of different ways to setup a functioning database, the optimal database was decided based on that which would allow the most straightforward implementation. Prior to making a more specific decision as to the platform to be used, a decision was made regarding whether a relational or non relational database would be used to save the digital signals the hardware was receiving. As scalability was prioritized in the solution and data was meant to be saved on a per user basis, a non relational database was seen as optimal. As the intention was to have data saved on a per-user basis, the key-value nature of non relational database allowed for a lookup model optimized for saving data for a particular user via an identifier.

Firebase was the database platform chosen due to it serving as a packaged solution which allowed for a flexible design that followed the document object model (DOM). This, alongside the fact that all data is saved in Javascript Object Notation (JSON), would allow multiple pieces of data to be saved easily under the user's unique identifier. Lastly, Firebase's database's real-time nature was an additional plus because it allowed for live data to be accessed by multiple devices at run time.

5.3 Hardware Interfacing

The previously mentioned analog-to-digital converters provided digital signals that the Raspberry Pi could handle, however, in order to make sense of the signals being generated via the communication protocol, many parts of the signal had to be accounted for and addressed. The complexities of proper addressing and bit accounting were needed for the chosen analog-to-digital converter. For this reason, Adafruit's ADS1X15 Python library was used. This allowed the software to make sense of the signals received and allowed for additional offset addressing in order to account for additional analog-to-digital converters.

5.4 Data Processing

Data IO and location computation were performed at a set time interval that would both allow for a simulated real time execution and keep the amount of time needed for the execution of one process from substantially delaying another. Should the process being carried out by one thread exceed the refresh rate, the next thread's execution would not be carried out successfully and an operating system (OS) error would occur. Such an error could halt the program completely. Upon testing, further covered in Appendix XX, it was found that the ideal delay time for all the running threads was 100 milliseconds. With that imposed delay, the execution of one thread would not detrimentally affect the execution of another thread.

Upon data collection and multiple test runs, both of which are explained further in Appendices F.1 (MATLAB angle code) and F.2 (MATLAB distance code), Equation

(4) and Equation (5) were derived. They were extracted from the empirically produced plots shown in Figure 5.4 and Figure 5.5 and those equations were used in the location determination. The angle plot clearly was not a function due to the zig-zag happening at the top-left of the plot, so it became necessary to forcibly fit a polynomial to the curve in order to approximate the equation. This polynomial is shown in Figure 5.6. In both cases, the input, shown as x, refers to the digital signal received by the Raspberry Pi. As mentioned in the ADC section, the output of the ADC, shown as x here, only ranges from 0 to 2047. The distance value produced is in centimeters (cm) and the angle is measure in degrees (°).

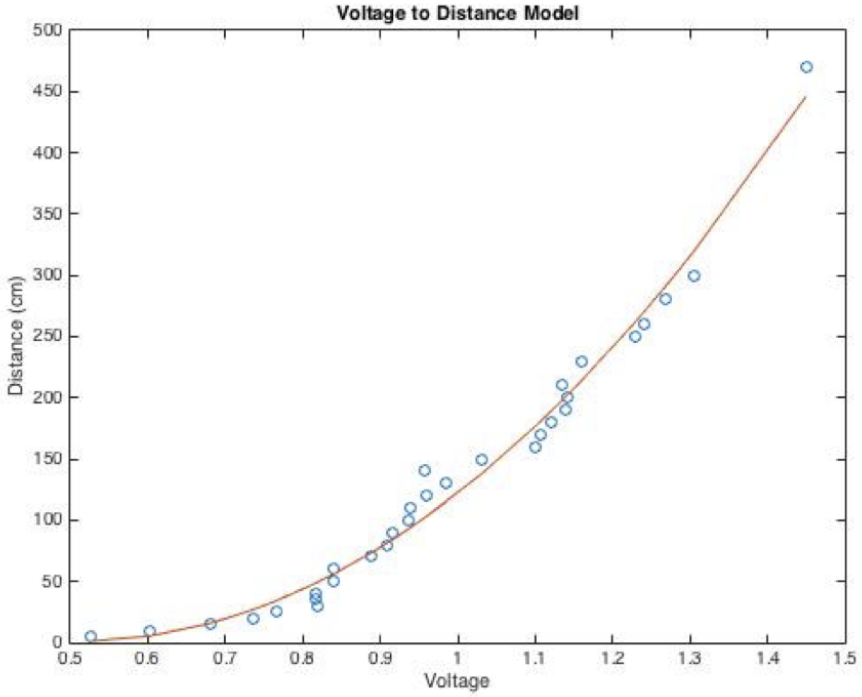


Figure 5.4: Empirical distance measurements.

$$\begin{aligned} Distance &= 503.509x^2 - 513.307x + 132.019 \\ Distance &= (503.509x - 513.307)x + 132.019 \end{aligned} \tag{4}$$

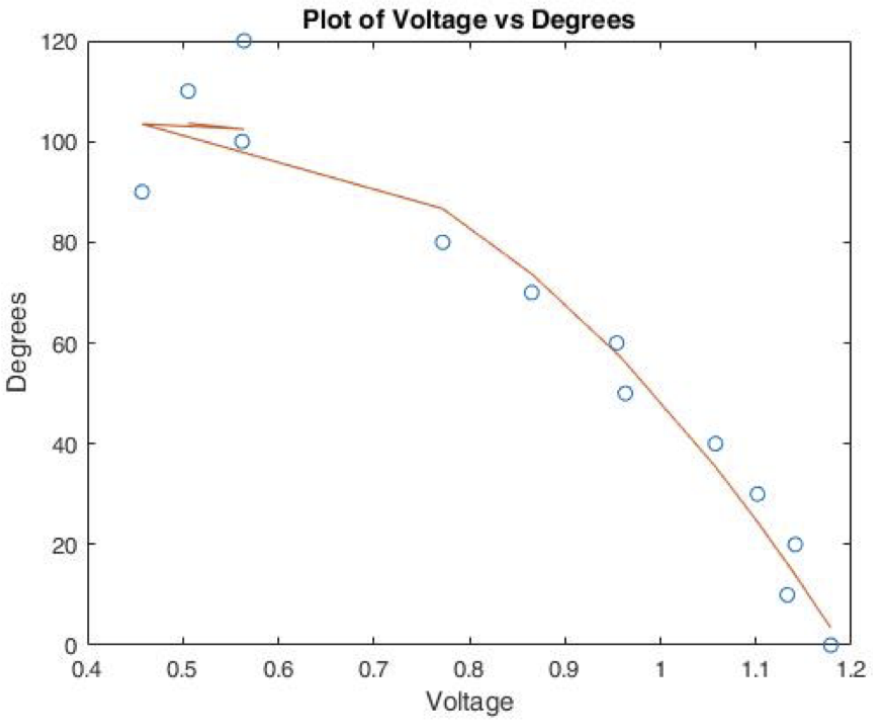


Figure 5.5: Empirical angle measurements.

$$\begin{aligned} Angle &= -209.936x^2 + 204.645x + 53.808 \\ Angle &= (-209.936x + 204.645)x + 53.808 \end{aligned} \tag{5}$$

In order to reduce the number of calculations performed at runtime, Horner's method was used. Horner's method essentially factors out terms that share power terms that have a degree greater than or equal to one. In both Equation (4) and Equation (5), the second degree and first degree term have one common x and so it is factored out in the optimized versions. In both cases, doing so equates to reducing the number of multiplication operations performed by one while keeping the number of addition operations the same. Since multiplying is one of the more intensive computations performed, right behind division, reducing the location computation by one multiplication is significant for optimization. Additionally, considering that a location is calculated at a rate

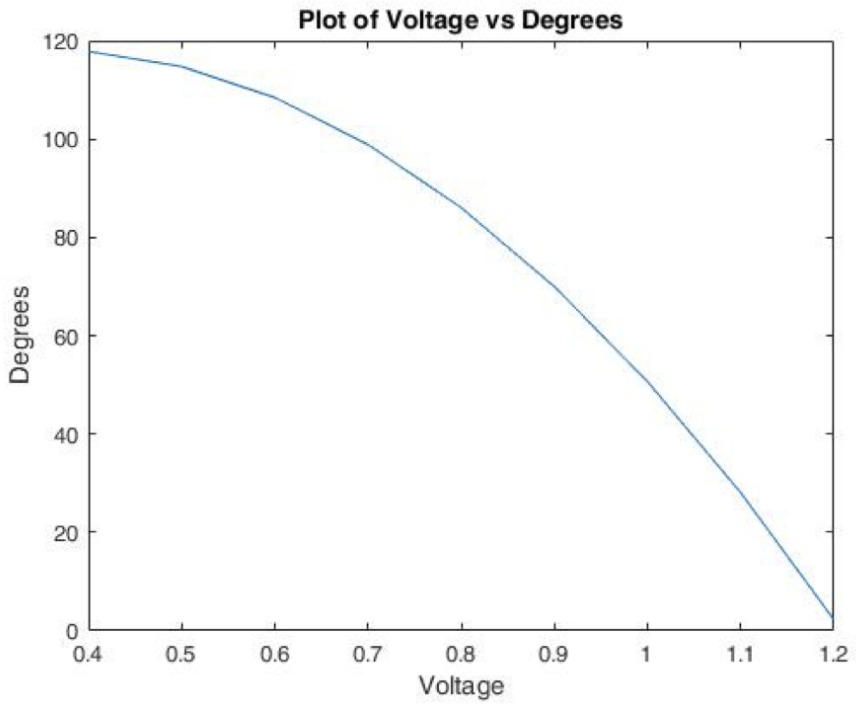


Figure 5.6: Angle plot force fit polynomial.

of 100 times per second, by optimizing the distance and angle equations, the number of performed operations every second is reduced by 100 multiplication operations.

5.5 Raspberry Pi Graphical User Interface

The application was optimized for the resistive touchscreen attached to the single-board computer running the application. This, though it allowed for facilitated user interaction, resulted in the limitation of solely being able to display movement along a single plane.

5.6 User Login

A login page is launched depending on an established Internet connection. Authorized user status is determined by Firebase authentication. The Internet connection is determined at the time that the application is launched. A connection is attempted to be

setup through a socket via Python's socket library. As a standard to check to, Google's host name (https://www.google.com) is used as the target host to try to establish a connection to. Once the host is attained, a connection is attempted. Should there be any exception thrown back, indicating a failed connection, the system continues in offline mode, else it continues in online mode.

5.6.1 Offline Mode

As a key feature of the system is to have Internet-independence, the primary functioning mode is that of an offline state. When the system is offline, authentication cannot be performed, and so the login page is entirely skipped and all location tracking information is localized solely to the hardware running the software application.

5.6.2 Online Mode

To address the goal of scalability in the number of location tracking users, a simple user email and password login was used. This aspect was done so that each user could have their unique login and a set of coordinates could be saved for each individual user. Upon authorization, via Firebase's email and password authentication, the user is redirected to the main page where their information is pushed to a real-time database under their user ID.

5.6.3 Primary Landing Page

In order to reduce the amount of clutter and minimize the amount of processing power needed to run the program alongside extended data IO uptime, the page's primary page of operation is minimal in features and only opts to include that which is function-critical. Figure 5.7 shows the minimal design of the actual graphical user interface. In designing the plot to show the location of the RF source a decision was made between using Cartesian and polar coordinates with their corresponding plot styles. Considering the processed data took the form of an angle and distance, it was decided that a polar graph more clearly and accurately relayed the information to the user.

Though it was determined that the plot would sufficiently show the position of the RF source, a decision was made on the technical aspect of the visual display and it determined that a more concrete form of position description would be needed for the execution.

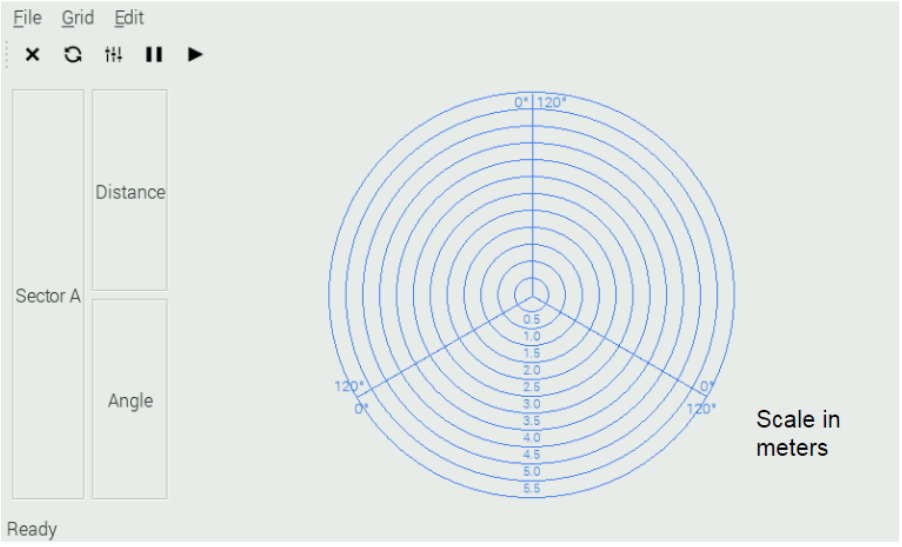


Figure 5.7: Main landing page.

5.7 Real-Time Database Platform

As mentioned previously, the data processing was done explicitly and solely on the hardware, in this case the Raspberry Pi, running the software application. However, as an additional aspect to broaden the prospective use of the digital signals being received by the hardware, all raw numeric values are posted up to the real-time database under an identifier unique to each user. While Figure 5.8 shows the list of users currently on the project's specific Firebase account, Figure 5.9 shows the saved location information of a user. By having a list of authorized users, a log in check can be performed on both the Raspberry Pi application and the web application.

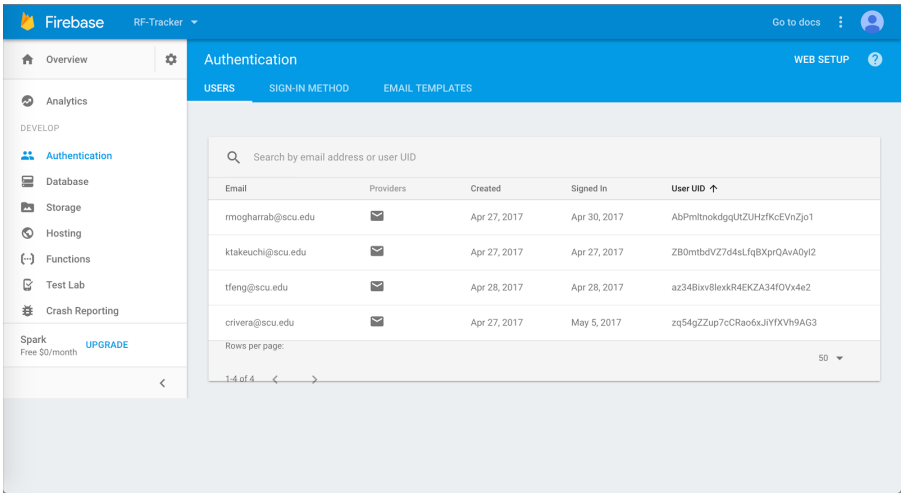


Figure 5.8: Firebase Authenticated User List.

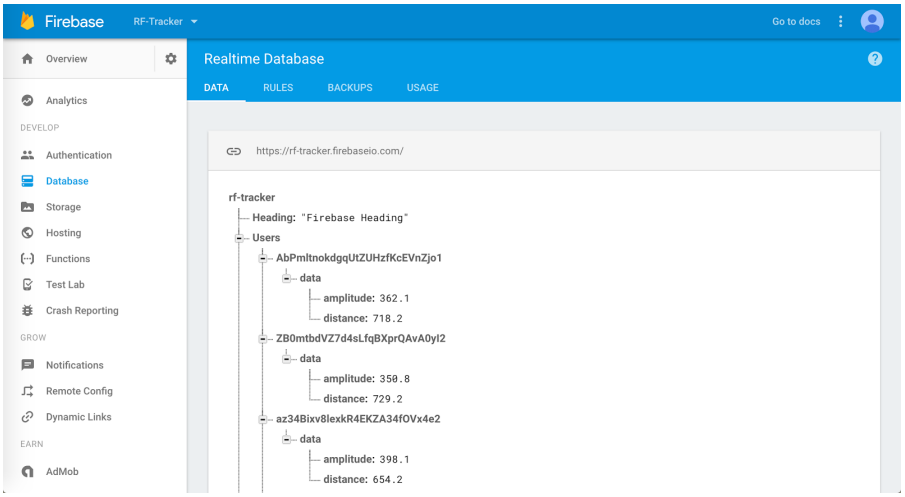


Figure 5.9: User saved location information.

5.8 Web User Interface

In order to test the viability of the location information saved onto the real-time database, a simple login and web user interface (UI) were used to show the way the location

information could be propagated to additional devices and platforms. As this was not critical to system functionality and rather an addition and proof to show the viability of this solution, the basis of the UI was derived from the Firebase template and a bootstrap theme.

5.8.1 Sign Up/Log In

In order to maximize the compatibility of the login with the setup Firebase database, the suggested simple login web page template offered by Firebase was used. A successful login resulted in the main menu being loaded. Since the login page provided directly by Firebase was used, no additional features were implemented and all of those on the original login page were retained. A point worth noting is that in the current version of the location tracking system, it is solely the web sign up that an authorized account can be made. Though this can be a limit in the account creation process, it assures the utmost security in the account creation as it is explicitly done in the way suggested by Firebase.

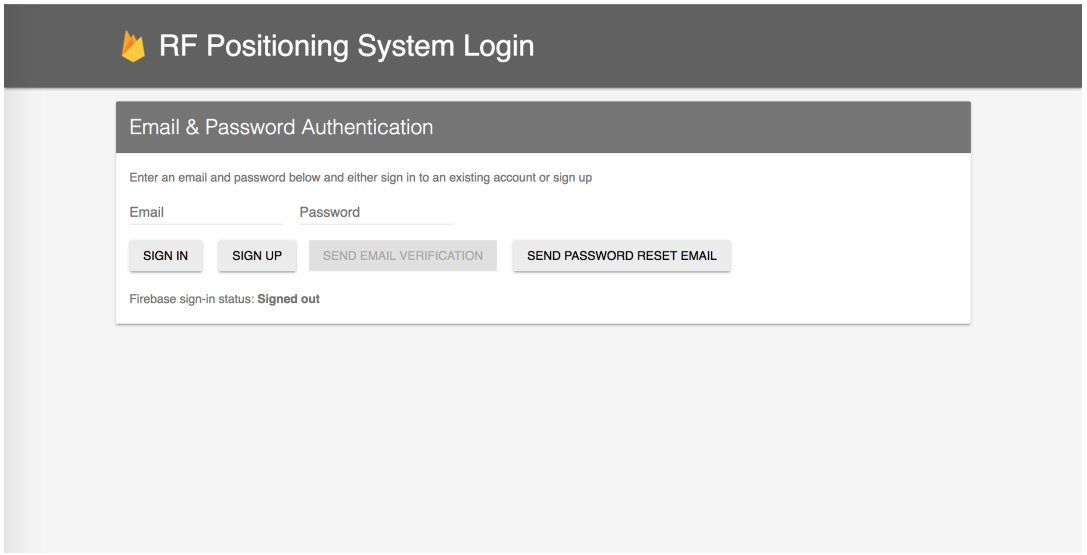


Figure 5.10: Login/Sign Up page

5.8.2 Real-Time Analytics

In order to continue with the minimalistic design used throughout the user interfaces, the main landing page of the web UI was meant to have a conservative layout, only focusing on the core information relevant to the positioning. The simplicity in the layout prioritized keeping the user from being confused by additional extraneous functionality. Due to the decision of prioritizing the visibility and readability on the portable screen, the polar plot's implementation was prioritized for the touchscreen and simply showing the values on the web UI was deemed sufficient.

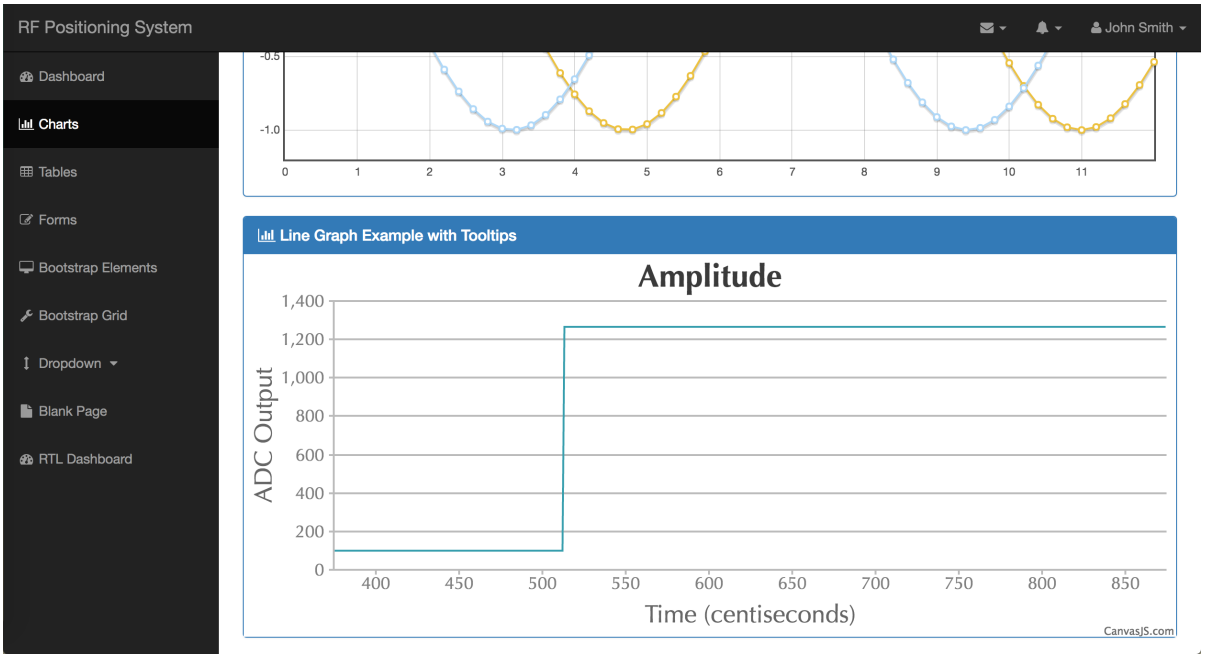


Figure 5.11: Live data plot.

The running plot was written using CanvasJS, a simple API that allows for both still and running plots. Since the data would be read at real-time, it was determined that their dynamic plots were ideal for data displaying.

6 System Testing

6.1 Antenna Base

System testing happened with many stages of improvement. At first, when testing began, we connected the SMA cables from the phase detector to the two Archimedian spiral antennas, then, using a paper under the antenna array that had degree measurements written down on it, we held the cables by hand and kept them 120° apart. This strategy obviously did not work.

Stage two of system testing consisted of 3D printing a base (Figure 6.1) that would hold the antennas 120° apart while leaving a hole in the center for the dipole whip antenna that would measure the relative distance. For the 3D print material, we chose PLA (polylactide) material because it has high surface hardness, it is easier to print, is biodegradable, and is abundantly available for 3D printing use. This second strategy also did not work because the dipole hole was too big and the SMA cables were too rigid, thus changing the spacial distance between the spiral antennas.

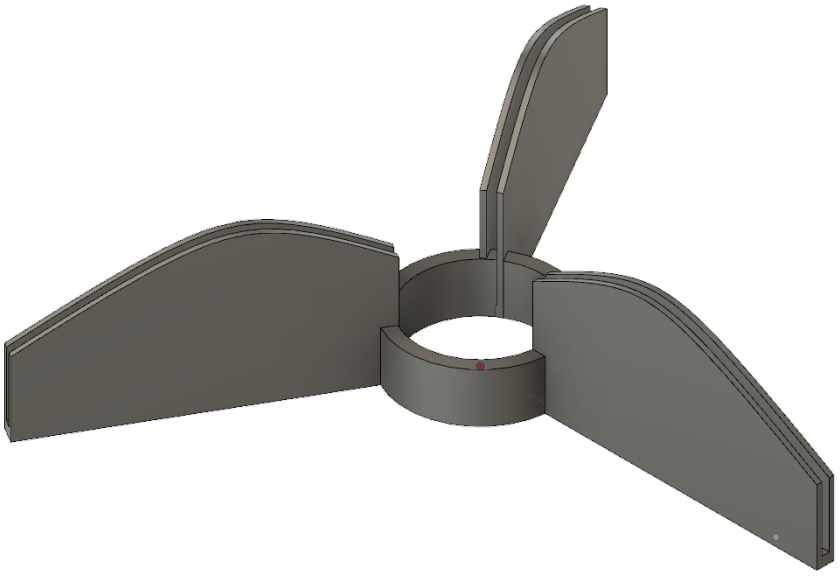


Figure 6.1: Antenna Base Version 1.

Finally, we created a second 3D printed holder (Figure 6.2) which had a base that provided rigidity, a much smaller hole in the middle, and no dipole hole (we decided to simply tape the dipole antenna against the two spirals to keep it secure).

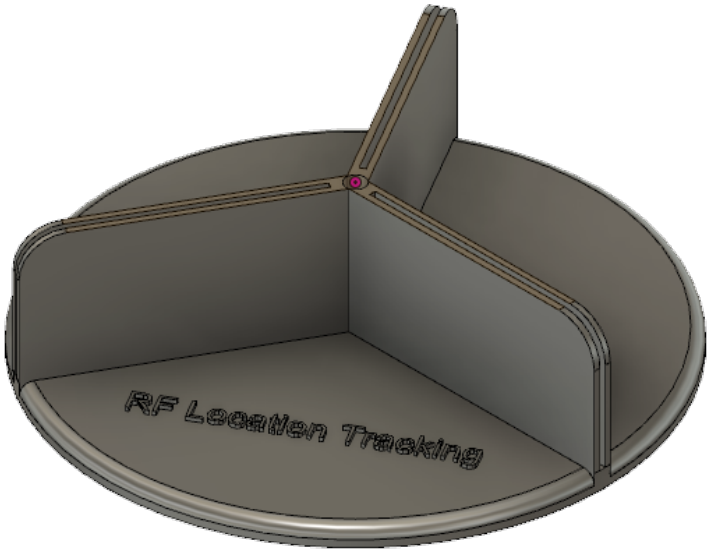


Figure 6.2: Antenna Base Version 2.

Taking advantage of the hole in the center of the base, we created a third 3D print model (Figure 6.3) that would plug into the bottom of the base and provide degree markings while we spun the antenna array so that we could compare the experimental direction readings read from the GUI to the actual direction the antenna array was facing. This information can be found in Appendix XX.

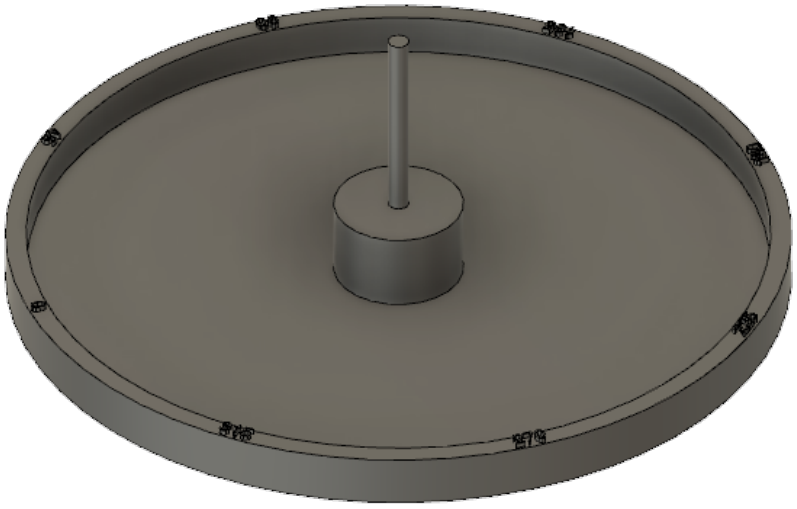


Figure 6.3: Spinning Base.

These bases were all designed using the Autodesk Fusion 360 CAD tool

6.2 Location Tracking Testing Method

Two types of RF sources were used in testing the location tracking system. A Hewlett Packard E4432B signal generator, shown in Figure 6.4 is the first RF source used. The advantage of using the E4432B signal generator is that it outputs a signal at constant power level, but the signal generator can only output a plane wave that does not carry any information. The signal generator also has the issue of being large and heavy, so it is not practical to test it, considering that it cannot be put on a drone. For this reason, using the signal generator as an RF source to track serves as a good test for an intial proof of concept, but not a final test.

In terms of the stationary setup used for this system test, a dipole antenna was plugged

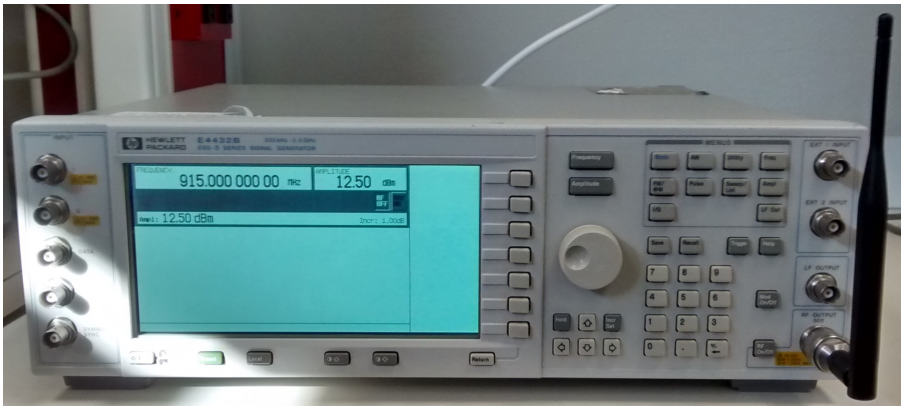


Figure 6.4: Hewlett Packard E4432B.

into the signal generator's output port. The signal generator outputs a 915MHz square wave at 12.5 dBm power. The center of the location tracker was placed 3 meters away from the signal generator. Since the system determines the location of the RF source relative to the tracker, instead of moving the signal generator around, the tracker was being moved for this test.

The other RF source used for system testing is a portable transmitter built with an Arudunio Uno and a RFM95W LoRa tranceiver. Figure 6.5 shows the portable transmitter. This transmitter outputs a modulated signal at 915MHz. Since the signal is modulated, information can be sent. This transmitter is a more accurate representation of the telemetry modules commonly used on drones, so it serves as a better test. However, the problem with this handheld transmitter is that its output signal has a fluctuating power level, which makes it more difficult to recitfy when processing the signal and mapping it to a distance or direction value.

For the portable setup used in this system test, the handheld transmitter outputs a signal that carries an integer at 12.5 dBm. The portable transmitter is then moved around relative to the antenna array and the relative position is viewed on the GUI with a legend marking the scale of the distance (in meters) and the direction (in degrees).

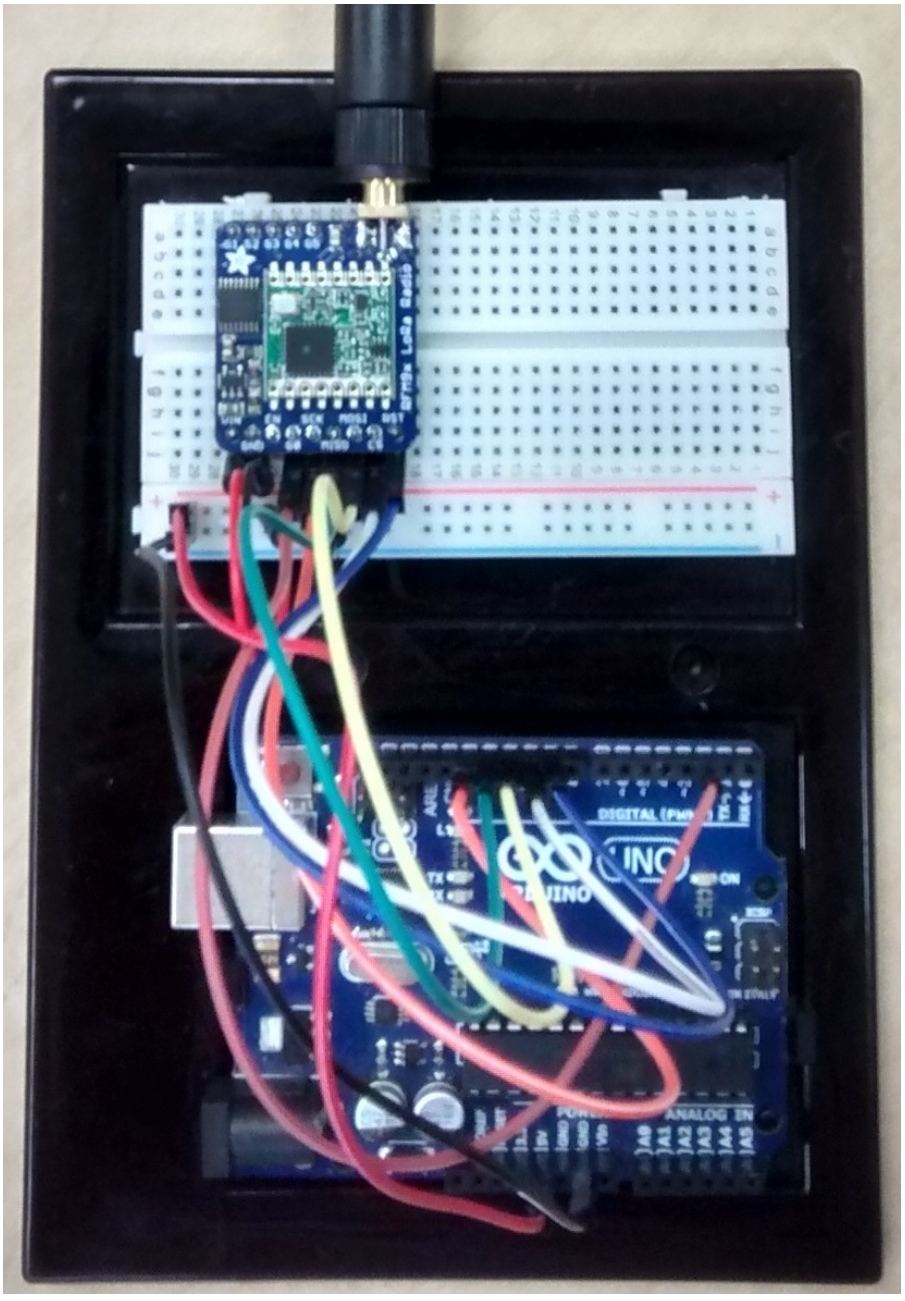


Figure 6.5: RFM95W LoRa Tranceiver.

6.3 Test Results

The tracker was able to determine the location of the Hewlett Packard E4432B signal generator fairly accurately. The distance resolution was tested to be about 5 centimeters.

The maximum angular error was found to be about 1°. Due to the size of the microwave lab at Santa Clara University, we only had space to test the system up to about 6 meters. For this reason, the range of the system tracking a Hewlett Packard E4432B signal generator is tested to be 6 meters.

The tracking system was not able to determine the location of the portable transmitter as well. The portable transmitter cannot output signals at constant power level. Through a spectrum analyzer test, we found that the output power level of the portable transmitter fluctuates over time. Since the tracking system determines of the location of an RF source using received power levels, the fluctuation of output power from the portable transmitter causes error.

7 Professional Issues and Constraints

7.1 Ethical

To analyze the engineering ethics of our senior design project, we first aim to identify important stakeholders. Then, we will determine the most vulnerable party. This will help us judge the risks and rewards attributed to each stakeholder involved and to come up with solutions to minimize the risk and maximize the reward.

7.1.1 Stakeholders

The most prominent stakeholders for our senior design project at its current stage would be:

- The design team
- The advisors, assisting professors
- The university (Santa Clara University)
- The suppliers for our required parts
- The customers for the finished product
  - Civilians
  - Farmers
  - Military

The most vulnerable stakeholders in this case would be the civilians and farmers so the most ethical thing to do is to implement protections for these stakeholders in our product.

Being that this project aims to make relative location measurements much more precise, some ethical concerns arise and are listed in the table below.

Table 7.1: Project Stakes

Stakeholder	What's at stake?
Design Team	Time
Advisers/Professor/School	Time
Santa Clara University	Money
Civilians	Safety (new search and rescue system)
Farmers	Crops, reduces physically labor-intensive work

Table 7.2: Ethical Concerns

Concerns	Why?	Solutions?
Unwarranted tracking	The simple system allows for cheap, precise, low-to-mid range tracking	Propose government regulations
Interference/Signal jamming of the system	Compromise mission-critical functions and desired performance	Frequency-hopping system to deal with jamming

7.2 Social

7.2.1 User Benefit

Our product is a device that allows for location tracking in spaces, such as indoor and areas with low reception, where GPS does not function, or is not precise enough. It will provide location-tracking functionality while also being very easy to use because it does not require any external component setup such as wireless access points, radio cell towers, or GPS satellites.

- Accessibility and Usability Over WiFi
  - Currently, the only other way that location tracking can be done in places where GPS does not work is using WiFi. But WiFi-based positioning systems require a wireless access point. Wireless access points are not easily set up if they don't already exist in a particular space. Our product, on the other hand, eliminates the need for wireless access point setups. Our device will simply track the location of an RF source relative to the device itself in a

one-to-one fashion. So to track an RF source, the only device needed is our tracking device and nothing else.

By keeping the components needed for our system to a minimum, we not only minimize the amount of error introduced to the system itself, but also concurrently make the positioning system easy to use.

7.2.2 Health and Welfare

Since we are working with a radio frequency-based system, there are possibilities of causing health and social hazards if we are not careful with our system implementation, but most of these hazards can be eliminated if we made sure our system follows FCC guidelines.

- Human Exposure to Radio Frequency Electromagnetic Fields
  - Long term exposure to radiation with high enough intensity can negatively impact a person's health. For that reason, the FCC have guidelines in place for Human Exposure to Radio Frequency Electromagnetic Fields. In order for our product to be safe for human usage, our device must be able to track RF transmitters (at 915 MHz frequency) that have field strength and power density under the exposure limit set by the FCC.
- Radio Frequency Interference
  - We also have to make sure our device does not cause interference with other radio frequency system. Our device must operate at frequencies that are allowed by the FCC. We also have to make sure our device does not produce any harmonics at unwanted frequencies that could potentially violate FCC regulations.

7.3 Economic

7.3.1 Operational Cost

Our project was given about $900 in funding, of which we have used a little more than half. Compared to many other well-funded projects in the world, ours needs much less capital to operate. Also, because this is simply a prototype, it costs much more than the actual final product will when produced in bulk.

The current cost of operation for our prototype with an itemized list of resources and their purpose is as follows:

- Raspberry Pi ($39.99): Compute input information and display on GUI
- Raspberry Pi 7" Touchscreen Display ($59.99): GUI
- Raspberry Pi Display case ($24.99): Holds Pi and display in compact package
- 12-bit Analog-Digital Converter ($9.99): Converts a voltage to a bitstream for Pi to read
- Jumper Cables ($9.99): Includes 200 jumper cables for prototyping
- Three Archimedian Spiral Antennas (around $100 each): Centerpiece of project

7.3.2 Selling Price Target

As shown in the itemized list above, the main cost is in the GUI ($125 total), which was mostly setup for our personal testing ease. The final product will be a mass-produced and miniaturized antenna structure which should cost a fraction of the $100 per antenna that it currently costs, thus making our product economically viable for consumer and corporate use.

7.3.3 Potential Targeted Customers

- Hobbyists

  - Hobbyists will find value in our product because the final package will be a modular implementation that would make it very simple for hobbyists to plug into a microcontroller and do with it what they want. Because of the low power consumption of the system, it would also be easy to implement into a personal project without drastically draining batteries or ramping up electricity bills.

- Swarm robotics companies
  - Because swarm robotics is slowly gaining momentum, companies that are in the business must be able to tell the relative locations among the members of their swarm. This is where our product truly becomes invaluable. Because a swarm normally consists of hundreds of units, it will make our product valuable enough to this particular user that it will support continued delivery of our product over a long period of time.

- Military
  - The military could use our product by implementing it into their long-range weaponry. If long-range drone strikes have a better idea of where to land, geographically, they will be much better-targeted and avoid taking innocent lives.

- IOT stores
  - Recently, there has been a surge in implementing IOT into many things such as smart labels in stores that can tell when an item has been misplaced or stolen, or refrigerators that can tell when you run out of groceries. Our product could help such causes by being the crux of these IOT systems. This would also be economically viable in consistently delivering our product because there will always be products in stores and we can put our device on all of them.

7.4 Science, Technology, and Society

The world, especially as more years go by, has been shaped by science and technology. Thus, this project follows that trend by focusing on the societal impact that it can

offer. A society can function at peak efficiency when the threat of war is minimized. To realize that end, the location tracker utilizes its technology to provide feelings of safety for society. It is calming and reassuring to know that if foreign powers were to wage an attack, American technology could track the attack and stop it from happening, protecting its inhabitants.

7.5 Civic Engagement

This project pertains to the community greatly due to the fact that it is being developed in order to solve two community-related problems:

- Vastly improve drone-to-drone relative location tracking for commercial purposes such as drone delivery
- Provide a reliable method for national security agencies to track rogue drones that are attempting to do harm upon the American population

To that end, this project combines the aggregate knowledge, skills, values, and motivation of its team members to solving the aforementioned problems.

7.6 Manufacturability

We've considered fabricating antennas with copper sheets and FR4 substrate using the milling machine on campus. For 3D printing, we considered using ABS plastic instead of PLA.

Fabricating the antennas ourselves would result in a waste of materials. The method of using the milling machine is to cut the desired shape out of copper sheets. The unwanted copper that gets cut away will be wasted. With the shape of our antenna (spiral), the area that needs to be cut away from a copper sheet is larger than the area of the antenna itself. By using printed antennas instead, we would not be wasting copper. We would only use the amount of copper needed to fabricate the antenna without the need to cut away anything to create waste.

Our decision to go with PLA over ABS was supported by the fact that PLA is derived from plants, and therefore, biodegradable. ABS is not. We decided to buy printed antennas instead of designing and fabricating them ourselves to save time and money. The effort that we would need to put into fabricating antennas ourselves would delay our entire project schedule significantly. There is also no guarantee that the antennas we fabricate will work the way we want them to because antennas are very sensitive to build quality. We would have wasted a lot of our budget if the antennas we fabricated did not work and we needed to re-do them. From a design point of view, we chose PLA over ABS because PLA has a higher surface hardness than ABS. We needed the least flexible material in order to hold our antenna array against the very rigid cables.

7.7 Sustainability

There should be no maintenance cost for our product because the final product will be an IC chip package that cannot be changed and must be replaced if malfunctioning. Therefore, the cost of energy for maintaining the product is also nonexistent. The product itself can operate within the life expectancy of IC chips, which could be about 30 years. The batteries can be replaced so that will not affect the lifespan of the whole system. We are currently at the prototype stage, so upgradability isn't something we've considered. As far as competing products that could potentially replace our technology, we believe that it is very possible that they could emerge within 3 to 4 years due to the rapid growth in the RF field.

We have to point out that battery replacement will play a factor in the environmental impact before the end of the product life. There have been an increasing number of solutions to disposal and recycling of batteries from battery companies which will help reduce the negative environmental impact. A majority of our components contains plastic, mainly the PCB circuit board, antennas, and 3D printed models. Recycling of electronics is a process that isn't necessarily efficient, but in order for the electronic waste to be properly disposed of, one of two process is needed: shredding or dismantling. In shredding, large machine equipment must be used in order for the recyclable metals to be recovered from the the actual electronic device. Dismantling, however, allows for smaller components to be easily used again. This is a laborious process that isn't quite

precise and in order for components to be properly removed, precautions must be taken so that the ICs in the chips don't fry or burn out.

7.8 Environmental Impact

Our system uses components that are composed of metals and plastics. Aside from the deterioration or decomposition rate of these two materials, they are not naturally biodegradable, and as a result, would contribute to total community waste. Also, the disposal of components in our system would include batteries, which themselves contain highly corrosive chemicals that require special disposal methods. Not only is this inconvenient, but the proper disposal process of materials with corrosive chemicals is a process that still pollutes the environment. Considering that a number of integrated circuits were used in our system, we had to be very meticulous in choosing our components so that we avoid waste.

The following are the resources used in the implementation of the system.

- Plastic
  - In the development of the system prototype and testing, we used molds made of PLA. Upon scaling and if our system is mass produced, the material used and waste produced could change dramatically and exponentially, respectively. The antennas used in the system use a dielectric substrate. The on-board computer in the system uses a printed circuit board (PCB) and all integrated circuits (IC) are soldered on.
- Metal
  - Copper is used in our system's circuits and in the antennas. All forms of signal transmission, whether analog or digital, were done through the same material. This includes the different forms of I/O used such as SMA or I2C. As a protective barrier, the majority of the components in our system have some form of casing, which is primarily made of metal. This is particularly true for the phase detector, which has multiple delicate ICs that need to be protected from potential harmful extremities.

- Transistors/Diodes (Semiconductors)
  - Our system uses a multitude of ICs ranging from those packaged within our phase detector to the analog to digital converters (ADC) needed in order to interface our phase detector output to our Raspberry Pi. The phase detectors and RF power detector are at the core of the system and provide the data to generate a location.
- Battery
  - Batteries are used in both the transmitter and receiver modules. By making them easily removable from our system, we not only increase the modularity and flexibility of it, but keep the overall lifespan of the system independent of battery life.

8 Bill of Materials

The proposed budget accounts for all parts of the RF location tracking system. An additional portion of the budget has been set aside for parts replacement. This is for the case where the components were damaged on delivery or in use. An additional portion of the budget was set aside for the purchase of an android device. While the majority of testing and simulation of the Android application can be done through an emulator, the deployment and BLE connectivity requires a physical Android device.

Current projected funding sources only include Santa Clara University and Santa Clara University's School of Engineering. No external sponsors have been communicated for additional project funding. We are asking that Santa Clara University entirely fund our project with the intention that our system can be used both for education and real-world application. Lastly, if awarded, the Xilinx grants will allow us to implement our system onto drones and wearable technology.

The expected result from the project will be to have a fully functioning location tracking system that operates at 915 MHz and works accurately for a range of roughly 15 m. Given the short span of time allotted for the entirety of the project, the scope of the

Table 8.1: Proposed Budget List

Item	Description	Quantity	Base Price	Net Price
Raspberry Pi 3	Data processing	1	$50.00	$50.00
Raspberry Pi 3 touchscreen	User interface	1	$60.00	$60.00
Raspberry Pi 3 case	Compact package	1	$25.00	$25.00
AD8302 Phase Detector	Phase/amplitude difference detection	3	$35.00	$105.00
AD8317	RF power detector for power level reading	2	$15.00	$30.00
Parts Replacement	In case of damage	-	$100.00	$100.00
Shipping Cost	-	-	$50.00	$50.00
Total Cost	-	-	$826.00	$420.00

objective may need to be scaled back. Assuming that sufficient time is not allocated, the implemented system will be tested with solely one RF source. The software aspect of the location tracking system is expected to progress faster than the hardware and so the filtering and position mapping aspects of the microcontroller's algorithm are expected to be finished. The android application's main functionality should be to visually display the coordinate information and additional features and GUI utility will be added as needed and if time permits.

9 Conclusion

9.1 Academic Integration

The team that worked on this project pooled knowledge from a wide variety of many diverse classes in order to conglomerate them into the results discussed in this paper. As follows with the technical structure of the project, the classes that proved invaluable for the completion of this project include those from the microwave series, the electronics series, and a series of computer engineering courses.

From the microwave series, the team drew much knowledge from ELEN 105: Electromagnetics II and ELEN 706: Microwave Circuit Analysis & Design. Being that this project relied predominantly on antennas at the front end, the team utilized these classes to design, analyze, and implement antennas properly into the system. These two classes, in conjunction, provided enough technical-level background material on the behavior of antenna systems for the team to be able to make educated decisions in the project.

From the electronics series, the proper skills were gained from ELEN 115: Electronic Circuits I. This class gave the team a great breadth of introductions to a wide variety of electronics and component configurations that allowed a user to get a desired output out of an electronic system. It was during this class that the team discovered and made the decision to use a phase detector in the design of the project. Without general exposure to the many commonly used circuits of today, the team would have remained ignorant of current technologies and would have had to reinvent the wheel and design a circuit that produces something that already exists.

Finally, from the computer engineering perspective, the classes that aided in the completion of this project were COEN 174: Software Engineering, COEN 177: Operating Systems, COEN 160: Object Oriented Design and Analysis, COEN 178: Introduction to Database Systems, and CSCI 166: Numerical Analysis.

- COEN 174: Software Engineering
  - Taught software management, report writing, and tools and methods used in software development

- COEN 177: Operating Systems
  - Allowed the team to understand thread priorities, concurrent processes, operating system (Raspbian) limitations. Having this knowledge was crucial in making the decision to use the Raspbian operating system as opposed to other choices available for Linux-based devices
- COEN 160: Object Oriented Design
  - Gave the team the skills to break up a problem into key components and identify software interactions and relations. An object oriented approach was taken in developing a software solution the problems presented in the project, such as defining sectors, creating a marker for the RF source, and providing a user menu
- COEN 178: Introduction to Database Systems
  - Introduced the team to common database management systems and gave more options in choosing a DBMS. The class highlighted the strengths and weaknesses of SQL and NoSQL databases and was instrumental in helping to choose the right one for our system
- CSCI 166: Numerical Analysis
  - Introduced the team to a number of methods used in data collection and how errors in data are both handled and minimized. Provided a better understanding of the definition of optimization and provided a plethora of ways to improve on the application's efficiency.

9.2 Future Work

In completing this project, we had many accomplishments and many lingering tasks remaining. If we, or a future group were to continue this project, there are multiple improvements and upgrades that can be made in order to bring the project to completion and fulfill our initial vision.

The initial goal for this project was to solve the problem of drones flying unregulated

around the skies. Thus, the project began with the goal of tracking the geographic position of drones. To that end, we created a modular antenna system that accurately tracks the relative location of an RF source in one sector. In this effort, we were successful, however, a future upgrade would include completing the other two sectors to allow our system to take advantage of all three sectors and encompass the full 360°of tracking.

Another upgrade that can be done on the project is to compartmentalize and minimize the separate components. For example, the phase detector takes up a large amount of space and the cables connected to and from it also take up space and can be the source of attenuation in our system. It would behoove us to recreate the system using RSSI (Received Signal Strength Indication) to minimize attenuation losses and keep the overall size small. Ideally, we would use a more simplistic, elegant, and minimalist approach to producing a small, modular package.

Finally, there is one last test that can be done to prove that our system operates as needed and works to sufficiently solve the problem of drone tracking errors. That is to actually track a flying drone and measure precise values that correspond to its relative separation from a set point. After implementing this and improving reliability and consistency through the use of a Kalman Filter, we are confident that our system can provide users with the most precise localization possible in the world today.

References

[1] M. Kim, T. Kubo, and N.Y. Chong, RF Power Detector for Location Sensing, Proc. Int. Conf. on Control, Automation, and Systems, 2005

[2] M. Kim, T. Tsunenori, and N.Y. Chong, Object Location Sensing Using Active RFID Systems, Proc. Int. Symposium on Robotics and Automation, 2004

[3] D.M. Pozar, Microwave and RF Wireless Systems, Wiley Text Books, 2000

[4] D.M. Pozar, Microwave Engineering 4th ed, Wiley Text Books, 2012

[5] W.L. Stutzman and G.A. Thiele, Antenna Theory and Design, John Wiley and Sons Ltd., 1999

[6] C.A. Balantis, Antenna Theory: Analysis and Design, Wiley Text Books, 1996

Appendices

A Quad-Band Cellular Antenna SMA CEL-0834 Spec Sheet

GSM QUAD-BAND ANTENNA

Part No.: ADH-151XSAXX

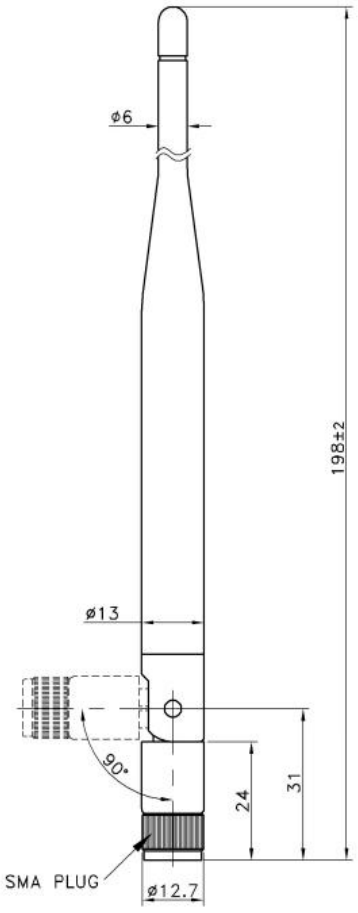
Index.

Item

Modification History:

Rev.	Date	Content
00	2010/11/18	

REV	ZONE	DESCRIPTION	ECN NO.	ENG	APPROVED	DATE



Electrical Properties	
Frequency Range	890~960 MHz/ 1710~1880 MHz
Impedance	50Ω
V.S.W.R.	≤ 2.5
Radiation	Omni
Gain	2dBi
Polarization	Vertical
Mechanical Properties	
Body Material	PU/TPE
Connector	Brass
Weight	23g (est.)
Operating Temp	−20℃ ~ +65℃

UNLESS OTHERWISE SPECIFIED TOLERANCES ON:
X ± 1
X.X ± 0.5
X.XX ± 0.25
ANG. ± 2.0°
THIRD ANGLE PROJECTION

DRAWN BY: JANE
DESIGNED BY: LICO
CHECKED BY: KYLIU
APPROVED BY: IRENE

THIS DRAWING CONTAINS INFORMATION THAT IS PROPRIETARY TO JOYMAX AND SHOULD NOT BE USED WITHOUT WRITTEN PERMISSION.

MATERIAL:
FINISH:
SCALE:1/1;UNIT:mm
DATE:2010/11/19
REV: 00

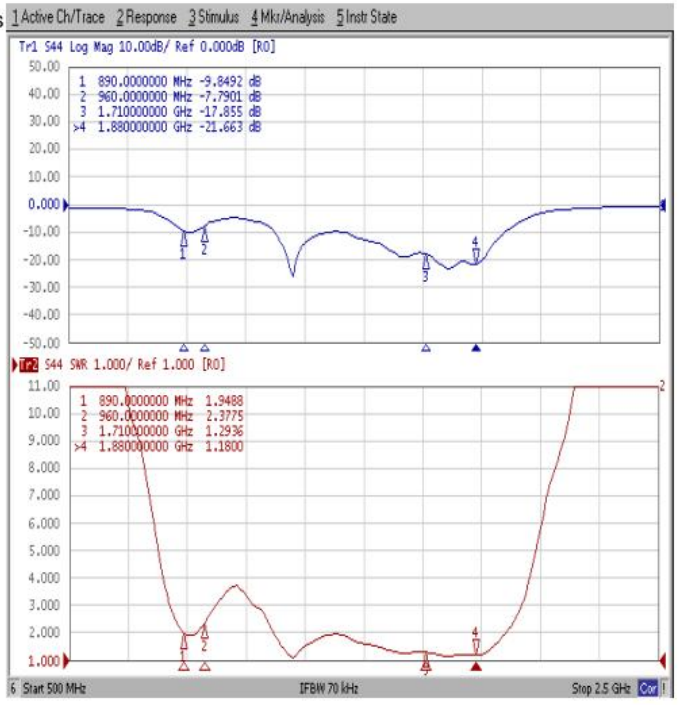
TITLE : #1511 Replacement Antenna
DRAW NO. :
PART NO. : ADH-151XSAXX

ADH Technology Co. Ltd.

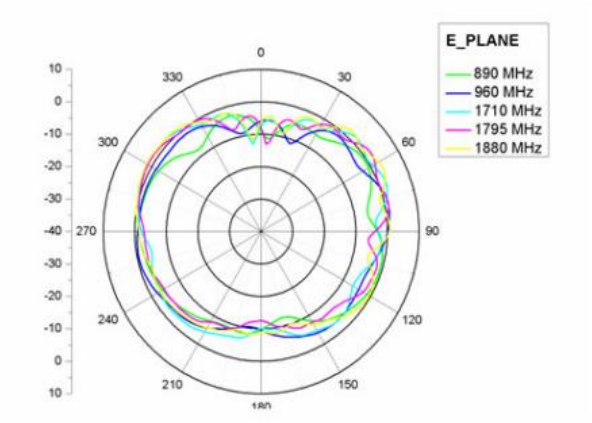
FM0414-02

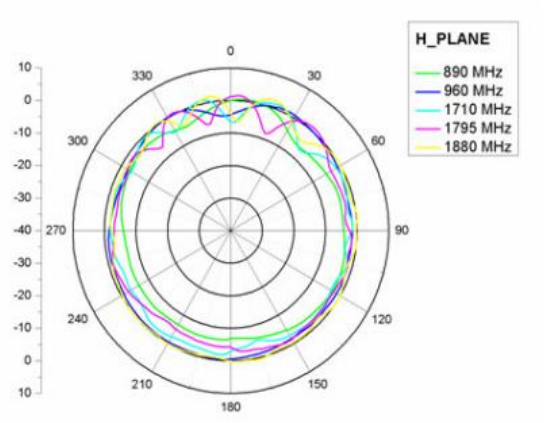
Return loss

V.S.W.R.



ANT1


E_PLANE
890 MHz
960 MHz
1710 MHz
1795 MHz
1880 MHz
0
30
60
90
120
150
180
210
240
270
300
330
10
0
-10
-20
-30
-40
-30
-20
-10
0
10


H_PLANE
890 MHz
960 MHz
1710 MHz
1795 MHz
1880 MHz
0
30
60
90
120
150
180
210
240
270
300
330
10
0
-10
-20
-30
-40
-30
-20
-10
0
10

Construction:

A) Center Conductor:
30 7/38 SPCW*
OD .012" ± .001"

B) Dielectric:
Extruded FEP
OD .033" ± .002"

C) Shield:
38 AWG SPC*
OD .051" Nom.

D) Jacket:
FEP – Brown Tint
OD .071" ± .004"

Electricals:

Impedance:	50 ± 2 Ohms
Capacitance:	32 pF/ft Max.
Velocity of Prop.:	70% Nom.
Cut off Frequency:	116 GHz
VSWR(.10 – 6.0 GHz):	1.20:1 Mean
Ramp Function:	
0.10GHz:	1.10:1
6.00GHz:	1.40:1

Physical Properties:

Weight per 1000 ft:	6.3 lbs Max.
Minimum Bend Radius:	.35"
Operating Temperature Range:	-55°C to 200°C
Conductor Break Strength:	4.6 lbs.

Attenuation:

0.10 GHz	14.0 dB/100ft.
0.40 GHz	28.2 dB/100ft.
1.00 GHz	45.0 dB/100ft.
2.00 GHz	64.4 dB/100ft.
2.45 GHz	71.6 dB/100ft.
3.00 GHz	79.7 dB/100ft.
4.00 GHz	92.7 dB/100ft.
5.00 GHz	104.3 dB/100ft.
6.00 GHz	115.0 dB/100ft

Connector		SMA
Specification Data	1） Impedance	50 ohm
	2） Frequency Range	0~6GHz
	3） V.S.W.R.	≦1.5
	4） Working Voltage	≦250 Vrms
	5） Dielectric Withstanding	≦670Vrms
	6） Voltage Insulation Resistance	≧2000 Mega ohm
	7） Contact Resistance	Center contact: 3.0 Milliohms (Max.) Outer contact: 2.0 Milliohms (Max.)
	8） Recommended coupling nut torque	4.0~8.8 in. lbs (0.45~0.99Nm)
	9） Coupling nut retention force	≧50 lbs (222N)
	10） Contact captivation force	≧5 lbs (22.2N)
	11） Durability (mating)	≧500 cycles
Environmental Data	1） Operating Temperature	－65℃～＋165℃
	2） Thermal Shock	MIL-STD-202,Method 107, Condition E
	3） Corrosion	MIL-STD-202,Method 101, Condition E
	4） Shock	MIL-STD-202,Method 213, Condition I
	5） Vibration	MIL-STD-202,Method 204, Condition D
	6） Moisture Resistance	MIL-STD-202,Method 106
Material Specifications	Material Data	Material
	1） Body	Brass
	2） Contact	Brass
	3） Insulator	Teflon or Delrin

B Empirically-collected Angle Data

C Empirically-collected Distance Data

D Analog Devices AD8302 Information

LF–2.7 GHz RF/IF Gain and Phase Detector

AD8302

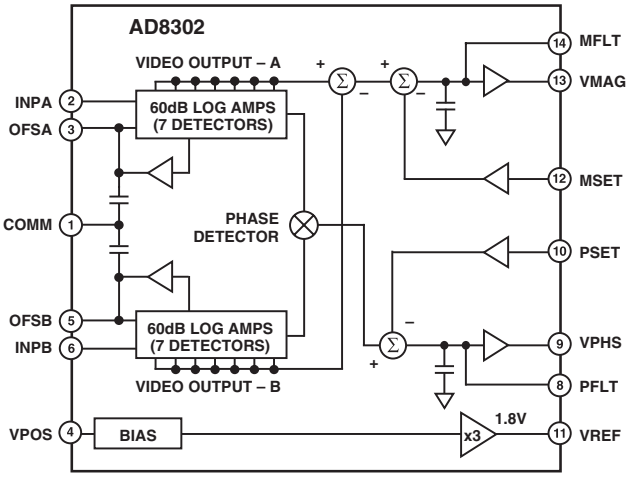
FEATURES

Measures Gain/Loss and Phase up to 2.7 GHz
Dual Demodulating Log Amps and Phase Detector
Input Range –60 dBm to 0 dBm in a 50 Ω System
Accurate Gain Measurement Scaling (30 mV/dB)
Typical Nonlinearity < 0.5 dB
Accurate Phase Measurement Scaling (10 mV/Degree)
Typical Nonlinearity < 1 Degree
Measurement/Controller/Level Comparator Modes
Operates from Supply Voltages of 2.7 V–5.5 V
Stable 1.8 V Reference Voltage Output
Small Signal Envelope Bandwidth from DC to 30 MHz

APPLICATIONS

RF/IF PA Linearization
Precise RF Power Control
Remote System Monitoring and Diagnostics
Return Loss/VSWR Measurements
Log Ratio Function for AC Signals

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD8302 is a fully integrated system for measuring gain/loss and phase in numerous receive, transmit, and instrumentation applications. It requires few external components and a single supply of 2.7 V–5.5 V. The ac-coupled input signals can range from –60 dBm to 0 dBm in a 50 Ω system, from low frequencies up to 2.7 GHz. The outputs provide an accurate measurement of either gain or loss over a ±30 dB range scaled to 30 mV/dB, and of phase over a 0°–180° range scaled to 10 mV/degree. Both subsystems have an output bandwidth of 30 MHz, which may optionally be reduced by the addition of external filter capacitors. The AD8302 can be used in controller mode to force the gain and phase of a signal chain toward predetermined setpoints.

The AD8302 comprises a closely matched pair of demodulating logarithmic amplifiers, each having a 60 dB measurement range. By taking the difference of their outputs, a measurement of the magnitude ratio or gain between the two input signals is available. These signals may even be at different frequencies, allowing the measurement of conversion gain or loss. The AD8302 may be used to determine absolute signal level by applying the unknown signal to one input and a calibrated ac reference signal to the other. With the output stage feedback connection disabled, a comparator may be realized, using the setpoint pins MSET and PSET to program the thresholds.

The signal inputs are single-ended, allowing them to be matched and connected directly to a directional coupler. Their input impedance is nominally 3 kΩ at low frequencies.

The AD8302 includes a phase detector of the multiplier type, but with precise phase balance driven by the fully limited signals appearing at the outputs of the two logarithmic amplifiers. Thus, the phase accuracy measurement is independent of signal level over a wide range.

The phase and gain output voltages are simultaneously available at loadable ground referenced outputs over the standard output range of 0 V to 1.8 V. The output drivers can source or sink up to 8 mA. A loadable, stable reference voltage of 1.8 V is available for precise repositioning of the output range by the user.

In controller applications, the connection between the gain output pin VMAG and the setpoint control pin MSET is broken. The desired setpoint is presented to MSET and the VMAG control signal drives an appropriate external variable gain device. Likewise, the feedback path between the phase output pin VPHS and its setpoint control pin PSET may be broken to allow operation as a phase controller.

The AD8302 is fabricated on Analog Devices' proprietary, high performance 25 GHz SOI complementary bipolar IC process. It is available in a 14-lead TSSOP package and operates over a –40°C to +85°C temperature range. An evaluation board is available.

REV. A

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329–4700 **www.analog.com**
Fax: 781/326-8703
© Analog Devices, Inc., 2002

AD8302* PRODUCT PAGE QUICK LINKS

Last Content Update: 02/23/2017

COMPARABLE PARTS

View a parametric search of comparable parts.

EVALUATION KITS

- AD8302 Evaluation Board

DOCUMENTATION

Application Notes

- AN-1040: RF Power Calibration Improves Performance of Wireless Transmitters
- AN-653: Improving Temperature, Stability, and Linearity of High Dynamic Range RMS RF Power Detectors
- AN-691: Operation of RF Detector Products at Low Frequency

Data Sheet

- AD8302: LF-2.7 GHz RF/IF Gain and Phase Detector Data Sheet

TOOLS AND SIMULATIONS

- ADIsimPLL™
- ADIsimRF

REFERENCE MATERIALS

Product Selection Guide

- RF Source Booklet

Technical Articles

- Design a Logamp RF Pulse Detector
- Detecting Fast RF Bursts using Log Amps
- Log Amps and Directional Couplers Enable VSWR Detection
- Make Precise Base-Station Power Measurements
- Measurement and Control of RF Power, Part I
- Measurement and Control of RF Power, Part II
- Measurement and Control of RF Power, Part III
- Measuring the RF Power in CDMA2000 and W-CDMA High Power Amplifiers (HPAs)
- Measuring VSWR and Gain in Wireless Systems

DESIGN RESOURCES

- AD8302 Material Declaration
- PCN-PDN Information
- Quality And Reliability
- Symbols and Footprints

DISCUSSIONS

View all AD8302 EngineerZone Discussions.

SAMPLE AND BUY

Visit the product page to see pricing options.

TECHNICAL SUPPORT

Submit a technical question or find your regional support number.

DOCUMENT FEEDBACK

Submit feedback for this data sheet.

This page is dynamically generated by Analog Devices, Inc., and inserted into this data sheet. A dynamic change to the content on this page will not trigger a change to either the revision number or the content of the product data sheet. This dynamic page may be frequently modified.

AD8302—SPECIFICATIONS (T_A = 25°C, V_S = 5 V, VMAG shorted to MSET, VPHS shorted to PSET, 52.3 Ω shunt resistors connected to INPA and INPB, for Phase measurement $P_{INPA} = P_{INPB}$, unless otherwise noted.)

Parameter	Conditions	Min	Typ	Max	Unit
OVERALL FUNCTION					
Input Frequency Range		>0		2700	MHz
Gain Measurement Range	P_{IN} at INPA, P_{IN} at INPB = –30 dBm		±30		dB
Phase Measurement Range	ϕ_{IN} at INPA > ϕ_{IN} at INPB		±90		Degree
Reference Voltage Output	Pin VREF, –40°C ≤ T_A ≤ +85°C	1.72	1.8	1.88	V
INPUT INTERFACE	Pins INPA and INPB				
Input Simplified Equivalent Circuit	To AC Ground, f ≤ 500 MHz		3‖2		kΩ‖pF
Input Voltage Range	AC-Coupled (0 dBV = 1 V rms)	–73		–13	dBV
	re: 50 Ω	–60		0	dBm
Center of Input Dynamic Range			–43		dBV
			–30		dBm
MAGNITUDE OUTPUT	Pin VMAG				
Output Voltage Minimum	20 × Log (V_{INPA}/V_{INPB}) = –30 dB		30		mV
Output Voltage Maximum	20 × Log (V_{INPA}/V_{INPB}) = +30 dB		1.8		V
Center Point of Output (MCP)	$V_{INPA} = V_{INPB}$		900		mV
Output Current	Source/Sink		8		mA
Small Signal Envelope Bandwidth	Pin MFLT Open		30		MHz
Slew Rate	40 dB Change, Load 20 pF‖10 kΩ		25		V/µs
Response Time					
Rise Time	Any 20 dB Change, 10%–90%		50		ns
Fall Time	Any 20 dB Change, 90%–10%		60		ns
Settling Time	Full-Scale 60 dB Change, to 1% Settling		300		ns
PHASE OUTPUT	Pin VPHS				
Output Voltage Minimum	Phase Difference 180 Degrees		30		mV
Output Voltage Maximum	Phase Difference 0 Degrees		1.8		V
Phase Center Point	When $\phi_{INPA} = \phi_{INPB} \pm 90°$		900		mV
Output Current Drive	Source/Sink		8		mA
Slew Rate			25		V/µs
Small Signal Envelope Bandwidth			30		MHz
Response Time	Any 15 Degree Change, 10%–90%		40		ns
	120 Degree Change C_{FILT} = 1 pF, to 1% Settling		500		ns
100 MHz	MAGNITUDE OUTPUT				
Dynamic Range	±1 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		58		dB
	±0.5 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		55		dB
	±0.2 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		42		dB
Slope	From Linear Regression		29		mV/dB
Deviation vs. Temperature	Deviation from Output at 25°C				
	–40°C ≤ T_A ≤ +85°C, $P_{INPA} = P_{INPB}$ = –30 dBm		0.25		dB
	Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = ±25 dB, P_{INPB} = –30 dBm		0.25		dB
Gain Measurement Balance	$P_{INPA} = P_{INPB}$ = –5 dBm to –50 dBm		0.2		dB
	PHASE OUTPUT				
Dynamic Range	Less than ±1 Degree Deviation from Best Fit Line		145		Degree
	Less than 10% Deviation in Instantaneous Slope		143		Degree
Slope (Absolute Value)	From Linear Regression about –90° or +90°		10		mV/Degree
Deviation vs. Temperature	Deviation from Output at 25°C				
	–40°C ≤ T_A ≤ +85°C, Delta Phase = 90 Degrees		0.7		Degree
	Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, Delta Phase = ±30 Degrees		0.7		Degree

Parameter	Conditions	Min	Typ	Max	Unit
900 MHz	MAGNITUDE OUTPUT				
Dynamic Range	±1 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		58		dB
	±0.5 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		54		dB
	±0.2 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		42		dB
Slope	From Linear Regression		28.7		mV/dB
Deviation vs. Temperature	Deviation from Output at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = P_{INPB} = –30 dBm		0.25		dB
	Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = ±25 dB, P_{INPB} = –30 dBm		0.25		dB
Gain Measurement Balance	P_{INPA} = P_{INPB} = –5 dBm to –50 dBm		0.2		dB
	PHASE OUTPUT				
Dynamic Range	Less than ± 1 Degree Deviation from Best Fit Line		143		Degree
	Less than 10% Deviation in Instantaneous Slope		143		Degree
Slope (Absolute Value)	From Linear Regression about –90° or +90°		10.1		mV/Degree
Deviation	Linear Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, Delta Phase = 90 Degrees		0.75		Degree
	–40°C ≤ T_A ≤ +85°C, Delta Phase = ±30 Degrees		0.75		Degree
Phase Measurement Balance	Phase @ INPA = Phase @ INPB, P_{IN} = –5 dBm to –50 dBm		0.8		Degree
1900 MHz	MAGNITUDE OUTPUT				
Dynamic Range	±1 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		57		dB
	±0.5 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		54		dB
	±0.2 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		42		dB
Slope	From Linear Regression		27.5		mV/dB
Deviation vs. Temperature	Deviation from Output at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = P_{INPB} = –30 dBm		0.27		dB
	Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = ±25 dB, P_{INPB} = –30 dBm		0.33		dB
Gain Measurement Balance	P_{INPA} = P_{INPB} = –5 dBm to –50 dBm		0.2		dB
	PHASE OUTPUT				
Dynamic Range	Less than ±1 Degree Deviation from Best Fit Line		128		Degree
	Less than 10% Deviation in Instantaneous Slope		120		Degree
Slope (Absolute Value)	From Linear Regression about –90° or +90°		10.2		mV/Degree
Deviation	Linear Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, Delta Phase = 90 Degrees		0.8		Degree
	–40°C ≤ T_A ≤ +85°C, Delta Phase = ±30 Degrees		0.8		Degree
Phase Measurement Balance	Phase @ INPA = Phase @ INPB, P_{IN} = –5 dBm to –50 dBm		1		Degree
2200 MHz	MAGNITUDE OUTPUT				
Dynamic Range	±1 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		53		dB
	±0.5 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		51		dB
	±0.2 dB Linearity P_{REF} = –30 dBm (V_{REF} = –43 dBV)		38		dB
Slope	From Linear Regression		27.5		mV/dB
Deviation vs. Temperature	Deviation from Output at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = P_{INPB} = –30 dBm		0.28		dB
	Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, P_{INPA} = ±25 dB, P_{INPB} = –30 dBm		0.4		dB
Gain Measurement Balance	P_{INPA} = P_{INPB} = –5 dBm to –50 dBm		0.2		dB
	PHASE OUTPUT				
Dynamic Range	Less than ±1 Degree Deviation from Best Fit Line		115		Degree
	Less than 10% Deviation in Instantaneous Slope		110		Degree
Slope (Absolute Value)	From Linear Regression about –90° or +90°		10		mV/Degree
Deviation	Linear Deviation from Best Fit Curve at 25°C				
	–40°C ≤ T_A ≤ +85°C, Delta Phase = 90 Degrees		0.85		Degree
	–40°C ≤ T_A ≤ +85°C, Delta Phase = ±30 Degrees		0.9		Degree
REFERENCE VOLTAGE	Pin VREF				
Output Voltage	Load = 2 kΩ	1.7	1.8	1.9	V
PSRR	V_S = 2.7 V to 5.5 V		0.25		mV/V
Output Current	Source/Sink (Less than 1% Change)		5		mA
POWER SUPPLY	Pin VPOS				
Supply		2.7	5.0	5.5	V
Operating Current (Quiescent)	V_S = 5 V		19	25	mA
	–40°C ≤ T_A ≤ +85°C		21	27	mA

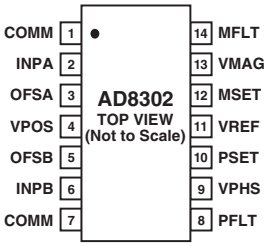
Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS[1]

Supply Voltage V_S . . . . . 5.5 V
PSET, MSET Voltage . . . . . V_S + 0.3 V
INPA, INPB Maximum Input . . . . . –3 dBV
Equivalent Power Re. 50 Ω . . . . . 10 dBm
θ_{JA}[2] . . . . . 150°C/W
Maximum Junction Temperature . . . . . 125°C
Operating Temperature Range . . . . . –40°C to +85°C
Storage Temperature Range . . . . . –65°C to +150°C
Lead Temperature Range (Soldering 60 sec) . . . . . 300°C

NOTES
[1]Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
[2]JEDEC 1S Standard (2-layer) board data.

PIN CONFIGURATION



PIN FUNCTION DESCRIPTIONS

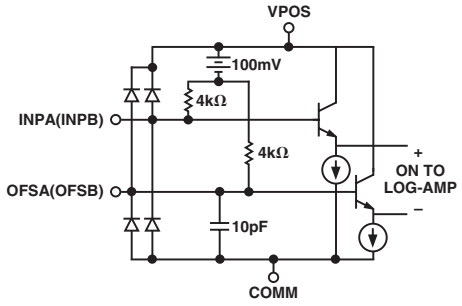
Pin No.	Mnemonic	Function	Equivalent Circuit
1, 7	COMM	Device Common. Connect to low impedance ground.	
2	INPA	High Input Impedance to Channel A. Must be ac-coupled.	Circuit A
3	OFSA	A capacitor to ground at this pin sets the offset compensation filter corner and provides input decoupling.	Circuit A
4	VPOS	Voltage Supply (V_S), 2.7 V to 5.5 V	
5	OFSB	A capacitor to ground at this pin sets the offset compensation filter corner and provides input decoupling.	Circuit A
6	INPB	Input to Channel B. Same structure as INPA.	Circuit A
8	PFLT	Low Pass Filter Terminal for the Phase Output	Circuit E
9	VPHS	Single-Ended Output Proportional to the Phase Difference between INPA and INPB.	Circuit B
10	PSET	Feedback Pin for Scaling of VPHS Output Voltage in Measurement Mode. Apply a setpoint voltage for controller mode.	Circuit D
11	VREF	Internally Generated Reference Voltage (1.8 V Nominal)	Circuit C
12	MSET	Feedback Pin for Scaling of VMAG Output Voltage Measurement Mode. Accepts a set point voltage in controller mode.	Circuit D
13	VMAG	Single-Ended Output. Output voltage proportional to the decibel ratio of signals applied to INPA and INPB.	Circuit B
14	MFLT	Low Pass Filter Terminal for the Magnitude Output	Circuit E

ORDERING GUIDE

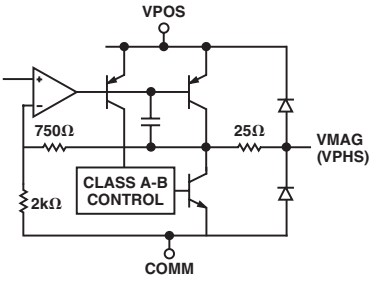
Model	Temperature Range	Package Description	Package Option
AD8302ARU	–40°C to +85°C	Tube, 14-Lead TSSOP	RU-14
AD8302ARU-REEL		13" Tape and Reel	
AD8302ARU-REEL7		7" Tape and Reel	
AD8302-EVAL		Evaluation Board	

CAUTION

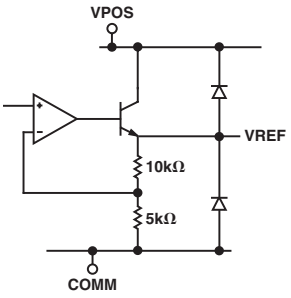
ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD8302 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.



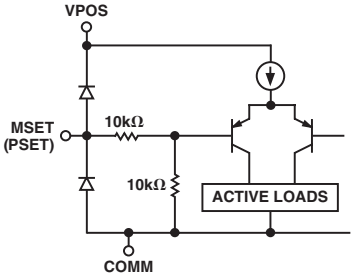
Circuit A



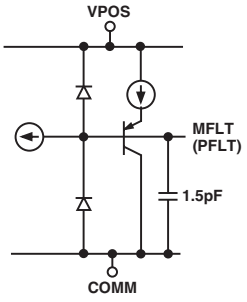
Circuit B



Circuit C



Circuit D

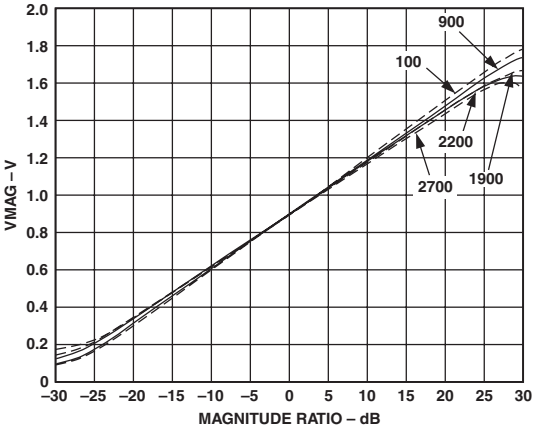


Circuit E

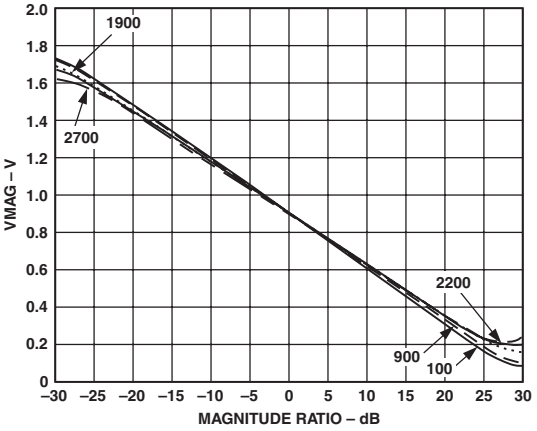
Figure 1. Equivalent Circuits

AD8302–Typical Performance Characteristics

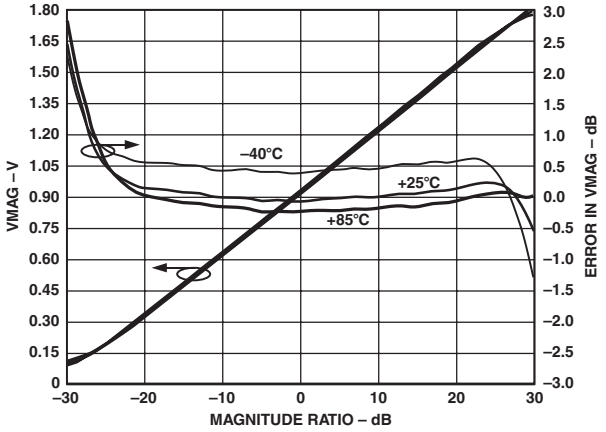
(V_S = 5 V, V_{INPB} is the reference input and V_{INPA} is swept, unless otherwise noted. All references to dBm are referred to 50 Ω. For the phase output curves, the input signal levels are equal, unless otherwise noted.)



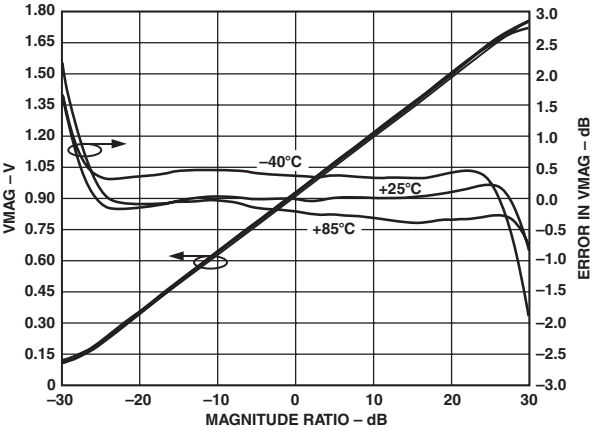
TPC 1. Magnitude Output (VMAG) vs. Input Level Ratio (Gain) V_{INPA}/V_{INPB}, Frequencies 100 MHz, 900 MHz, 1900 MHz, 2200 MHz, 2700 MHz, 25°C, P_{INPB} = –30 dBm, (Re: 50 Ω)



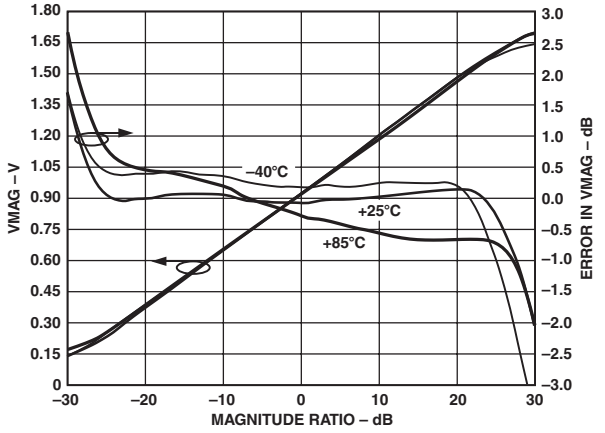
TPC 2. VMAG vs. Input Level Ratio (Gain) V_{INPA}/V_{INPB}, Frequencies 100 MHz, 900 MHz, 1900 MHz, 2200 MHz, 2700 MHz, P_{INPA} = –30 dBm



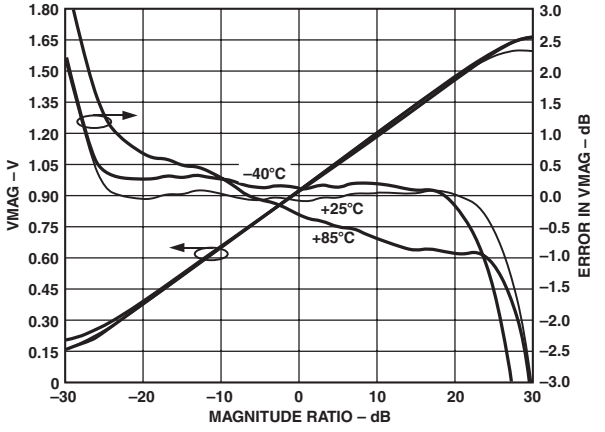
TPC 3. VMAG Output and Log Conformance vs. Input Level Ratio (Gain), Frequency 100 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



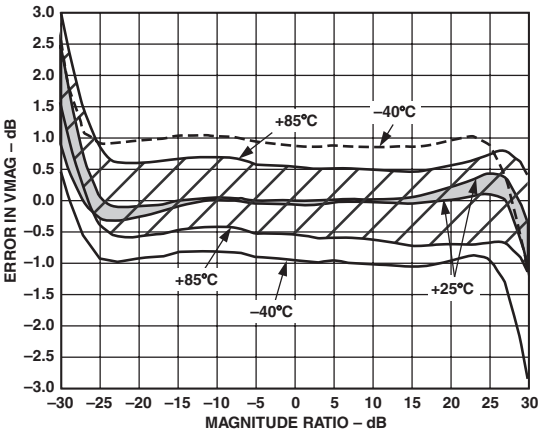
TPC 4. VMAG and Log Conformance vs. Input Level Ratio (Gain), Frequency 900 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



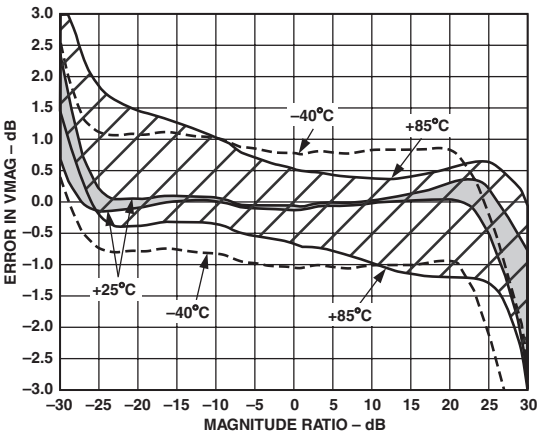
TPC 5. VMAG and Log Conformance vs. Input Level Ratio (Gain), Frequency 1900 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



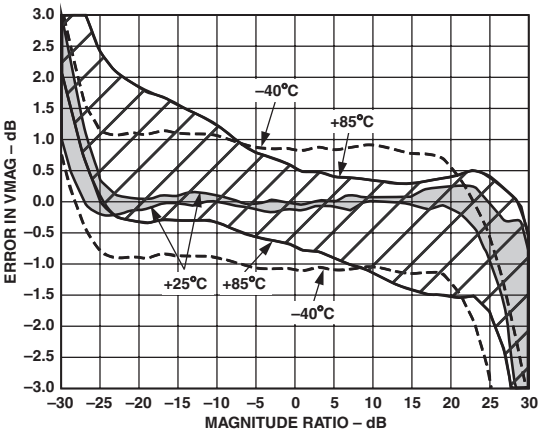
TPC 6. VMAG Output and Log Conformance vs. Input Level Ratio (Gain), Frequency 2200 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



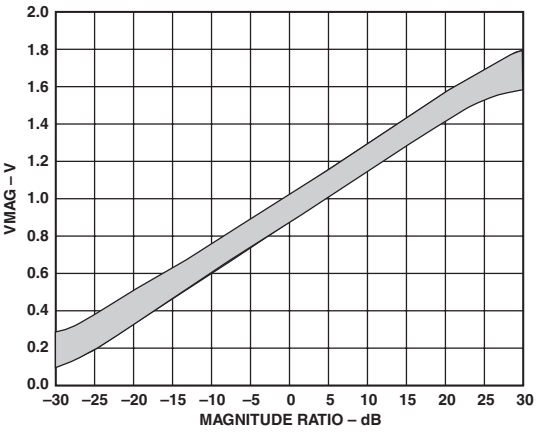
TPC 7. Distribution of Magnitude Error vs. Input Level Ratio (Gain), Three Sigma to Either Side of Mean, Frequency 900 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



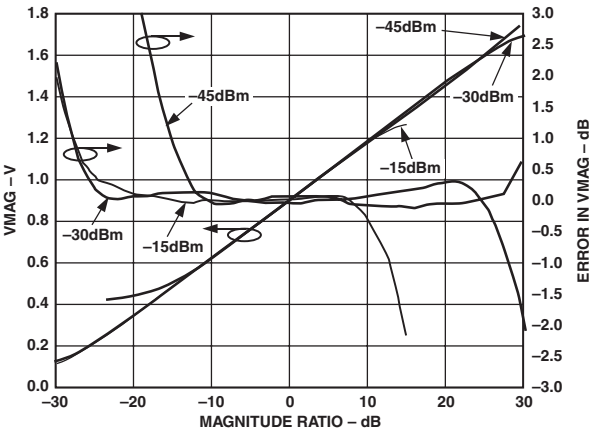
TPC 8. Distribution of Error vs. Input Level Ratio (Gain), Three Sigma to Either Side of Mean, Frequency 1900 MHz, –40°C, +25°C, and +85°C, Reference Level = –30 dBm



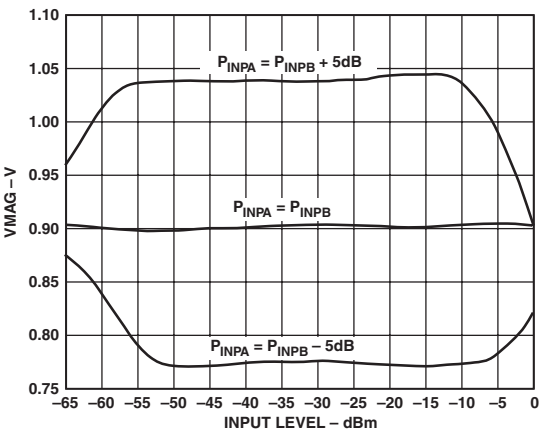
TPC 9. Distribution of Magnitude Error vs. Input Level Ratio (Gain), Three Sigma to Either Side of Mean, Frequency 2200 MHz, Temperatures –40°C, +25°C, and +85°C, Reference Level = –30 dBm



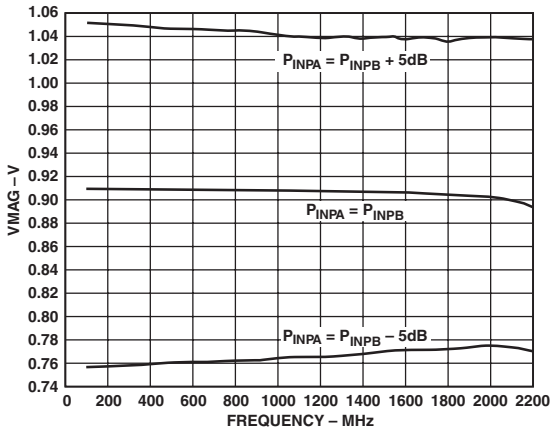
TPC 10. Distribution of VMAG vs. Input Level Ratio (Gain), Three Sigma to Either Side of Mean, Frequency 1900 MHz, Temperatures Between –40°C and +85°C, Reference Level = –30 dBm



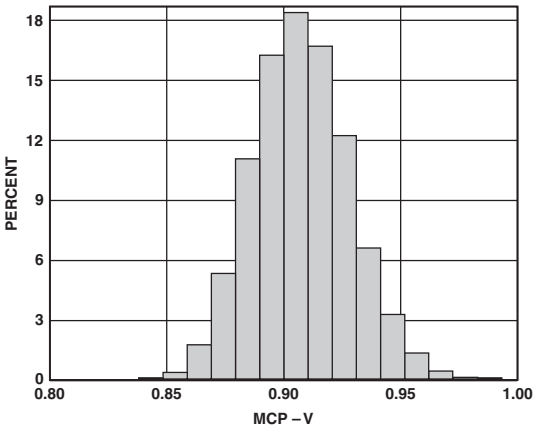
TPC 11. VMAG Output and Log Conformance vs. Input Level Ratio (Gain), Reference Level = –15 dBm, –30 dBm, and –45 dBm, Frequency 1900 MHz



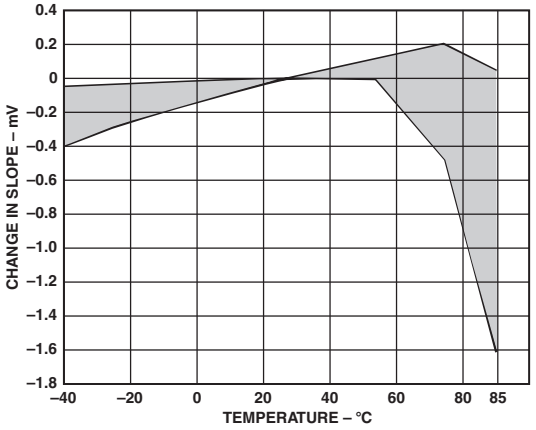
TPC 12. VMAG Output vs. Input Level for $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB} + 5$ dB, $P_{INPA} = P_{INPB} – 5$ dB, Frequency 1900 MHz



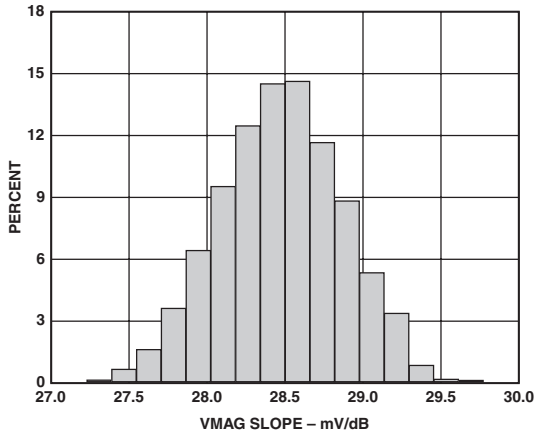
TPC 13. VMAG Output vs. Frequency, for $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB} + 5$ dB, and $P_{INPA} = P_{INPB} – 5$ dB, $P_{INPB} = –30$ dBm



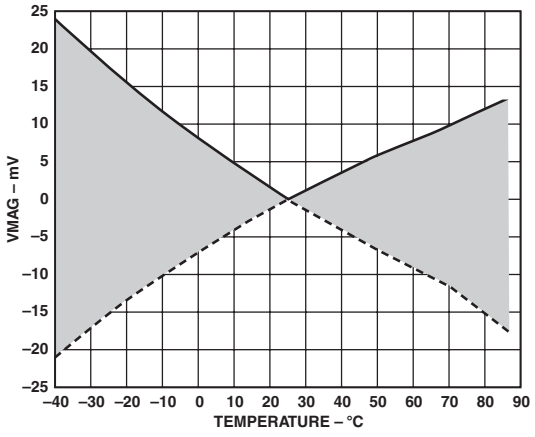
TPC 16. Center Point of Magnitude Output (MCP) Distribution Frequencies 900 MHz, 17,000 Units



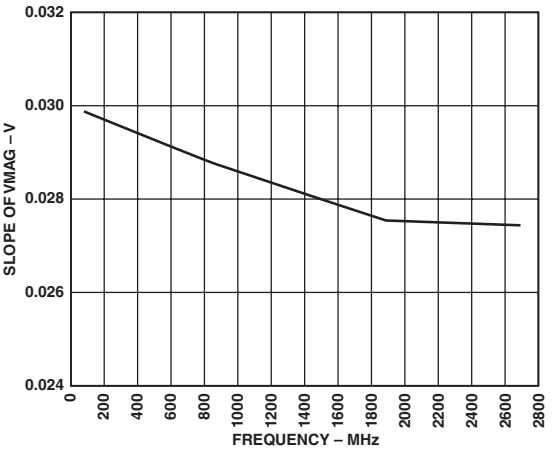
TPC 14. Change in VMAG Slope vs. Temperature, Three Sigma to Either Side of Mean, Frequencies 1900 MHz



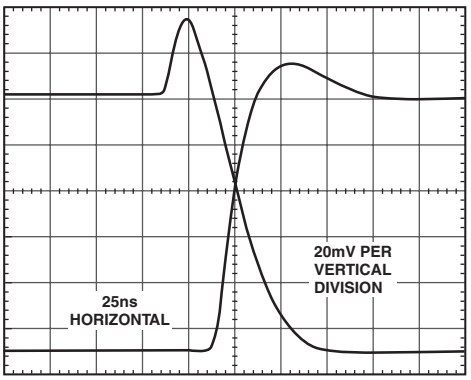
TPC 17. VMAG Slope, Frequency 900 MHz, 17,000 Units



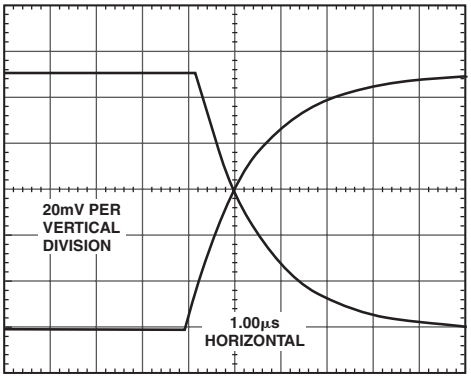
TPC 15. Change in Center Point of Magnitude Output (MCP) vs. Temperature, Three Sigma to Either Side of Mean, Frequencies 1900 MHz



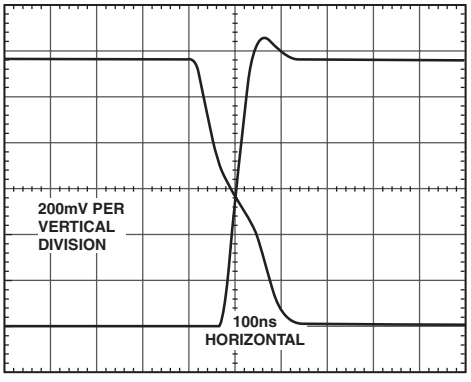
TPC 18. VMAG Slope vs. Frequency



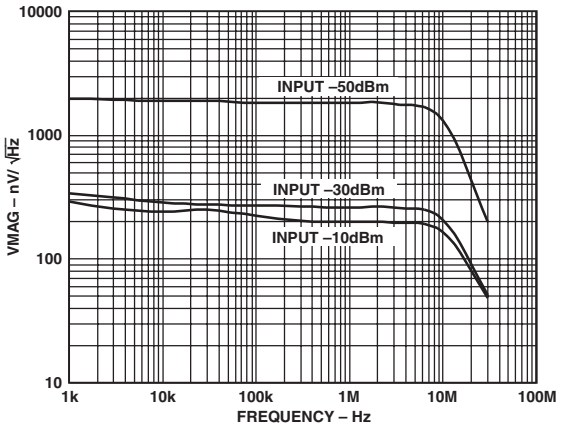
TPC 19. Magnitude Output Response to 4 dB Step, for P_{INPB} = –30 dBm, P_{INPA} = –32 dBm to –28 dBm, Frequency 1900 MHz, No Filter Capacitor



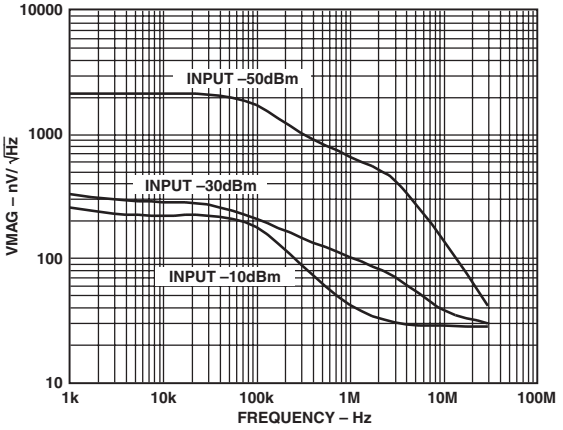
TPC 20. Magnitude Output Response to 4 dB Step, for P_{INPB} = –30 dBm, P_{INPA} = –32 dBm to –28 dBm, Frequency 1900 MHz, 1 nF Filter Capacitor



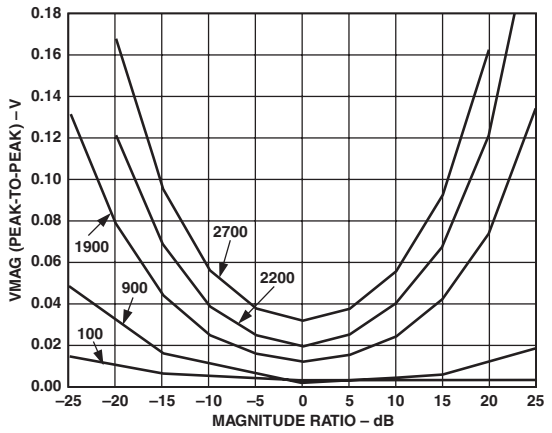
TPC 21. Magnitude Output Response to 40 dB Step, for P_{INPB} = –30 dBm, P_{INPA} = –50 dBm to –10 dBm, Supply 5 V, Frequency 1900 MHz, No Filter Capacitor



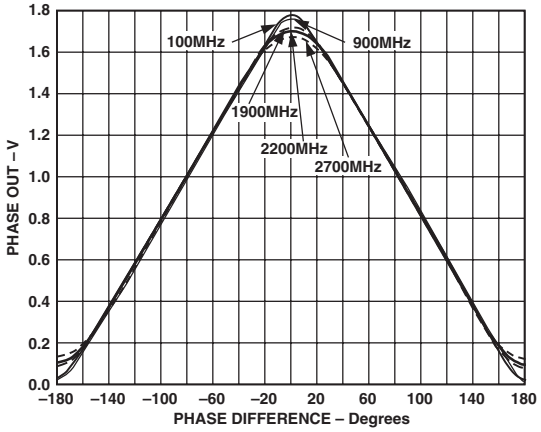
TPC 22. Magnitude Output Noise Spectral Density, P_{INPA} = P_{INPB} = –10 dBm, –30 dBm, –50 dBm, No Filter Capacitor



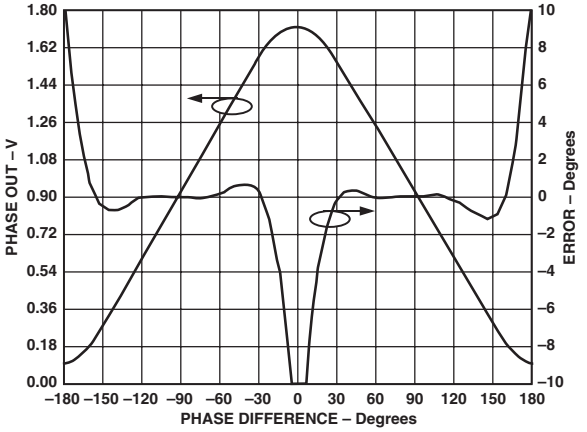
TPC 23. Magnitude Output Noise Spectral Density, P_{INPA} = P_{INPB} = –10 dBm, –30 dBm, –50 dBm, with Filter Capacitor, C = 1 nF



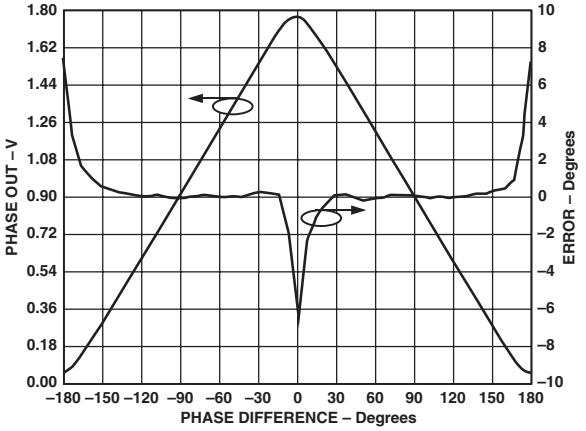
TPC 24. VMAG Peak-to-Peak Output Induced by Sweeping Phase Difference through 360 Degrees vs. Magnitude Ratio, Frequencies 100 MHz, 900 MHz, 1900 MHz, 2200 MHz, and 2700 MHz



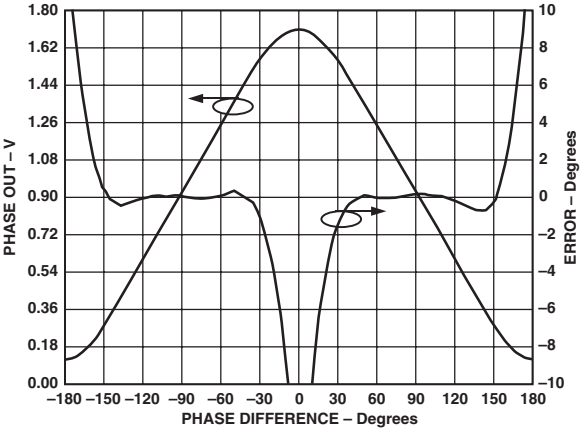
TPC 25. Phase Output (VPHS) vs. Input Phase Difference, Input Levels –30 dBm, Frequencies 100 MHz, 900 MHz, 1900 MHz, 2200 MHz, Supply 5 V, 2700 MHz



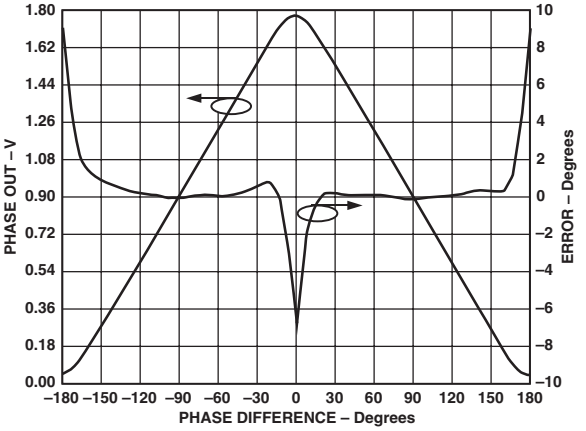
TPC 28. VPHS Output and Nonlinearity vs. Input Phase Difference, Input Levels –30 dBm, Frequency 1900 MHz



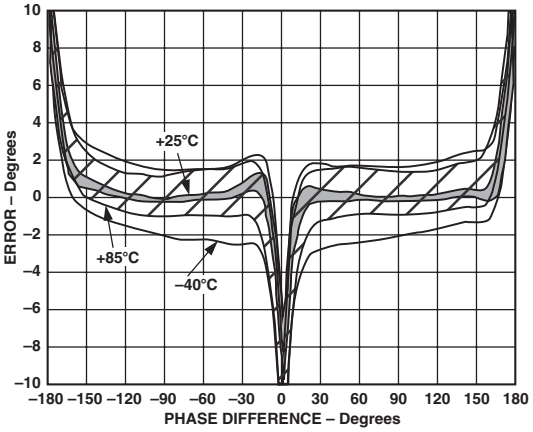
TPC 26. VPHS Output and Nonlinearity vs. Input Phase Difference, Input Levels –30 dBm, Frequency 100 MHz



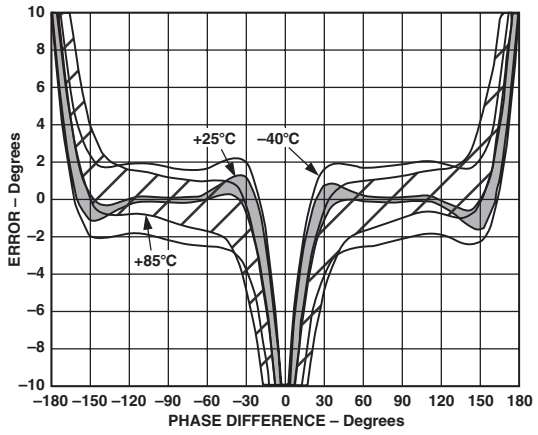
TPC 29. VPHS Output and Nonlinearity vs. Input Phase Difference, Input Levels –30 dBm, Frequency 2200 MHz



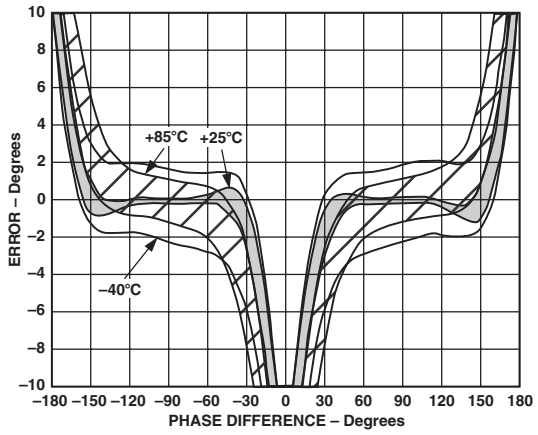
TPC 27. VPHS Output and Nonlinearity vs. Input Phase Difference, Input Levels –30 dBm, Frequency 900 MHz



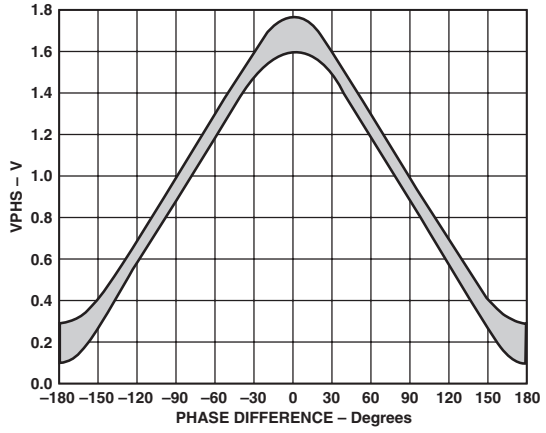
TPC 30. Distribution of VPHS Error vs. Input Phase Difference, Three Sigma to Either Side of Mean, Frequency 900 MHz, –40°C, +25°C, and +85°C, Input Levels –30 dBm



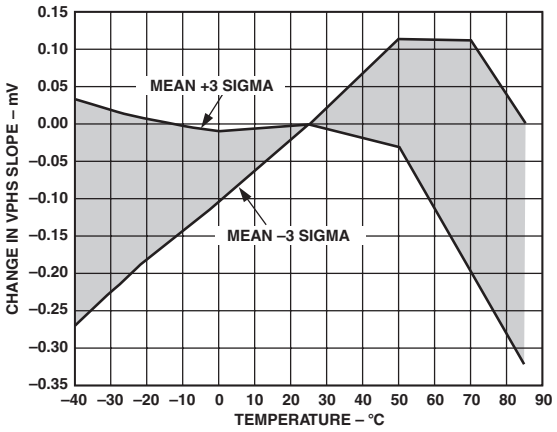
TPC 31. Distribution of VPHS Error vs. Input Phase Difference, Three Sigma to Either Side of Mean, Frequency 1900 MHz, –40°C, +25°C, and +85°C, Supply 5 V, Input Levels P_{INPA} = P_{INPB} = –30 dBm



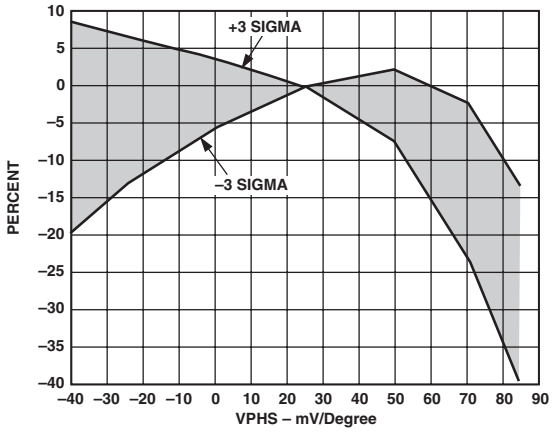
TPC 32. Distribution of VPHS Error vs. Input Phase Difference, Three Sigma to Either Side of Mean, Frequency 2200 MHz, –40°C, +25°C, and +85°C, Input Levels –30 dBm



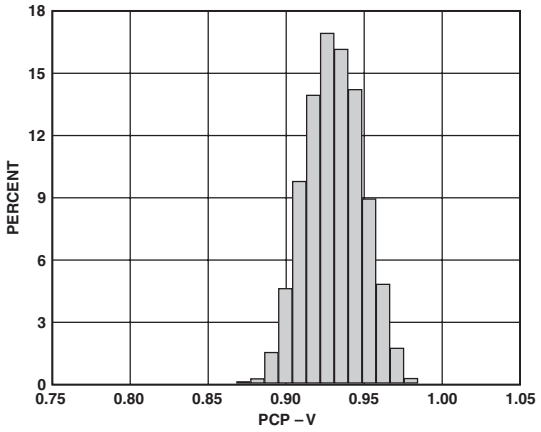
TPC 33. Distribution of VPHS vs. Input Phase Difference, Three Sigma to Either Side of Mean, Frequency 900 MHz, Temperature between –40°C and +85°C, Input Levels –30 dBm



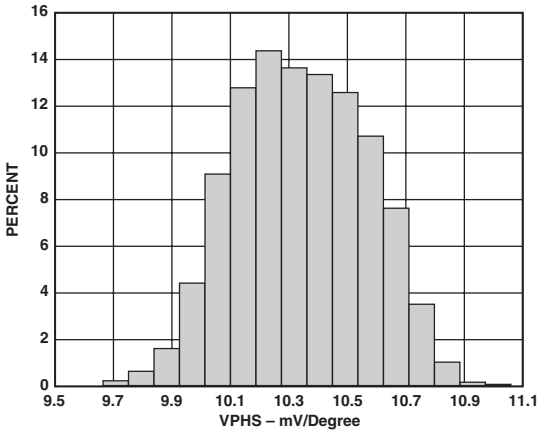
TPC 34. Change in VPHS Slope vs. Temperature, Three Sigma to Either Side of Mean, Frequency 1900 MHz



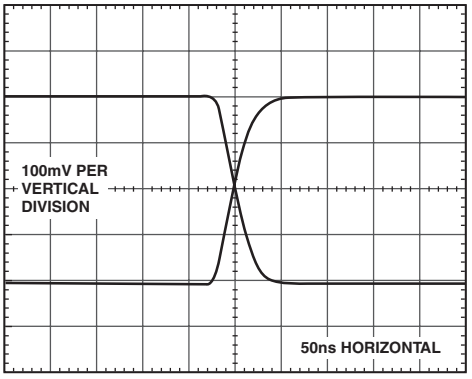
TPC 35. Change in Phase Center Point (PCP) vs. Temperature, Three Sigma to Either Side of Mean, Frequency 1900 MHz



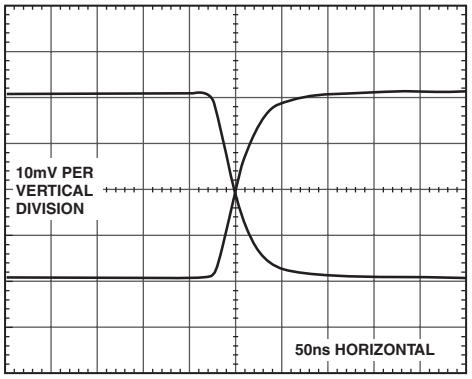
TPC 36. Phase Center Point (PCP) Distribution, Frequency 900 MHz, 17,000 Units



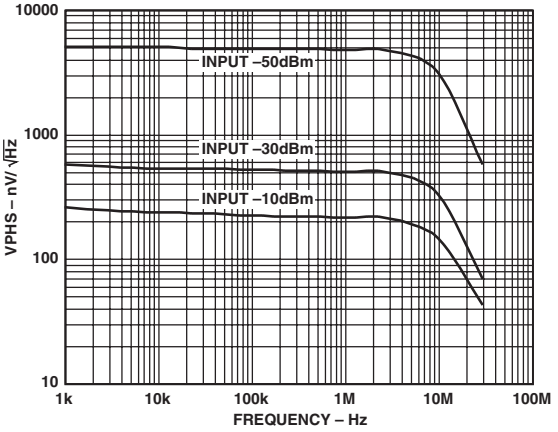
TPC 37. VPHS Slope Distribution, Frequency 900 MHz



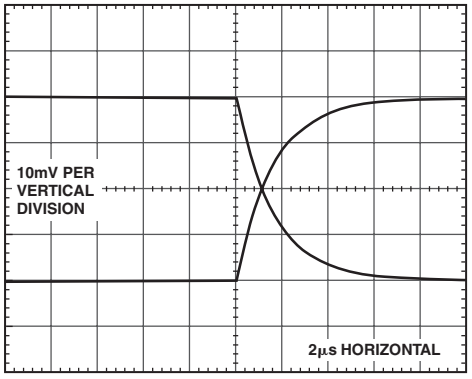
TPC 40. VPHS Output Response to 40° Step with Nominal Phase Shift of 90°, Input Levels P_{INPA} = P_{INPB} = –30 dBm, Frequency 1900 MHz,1 pF Filter Capacitor



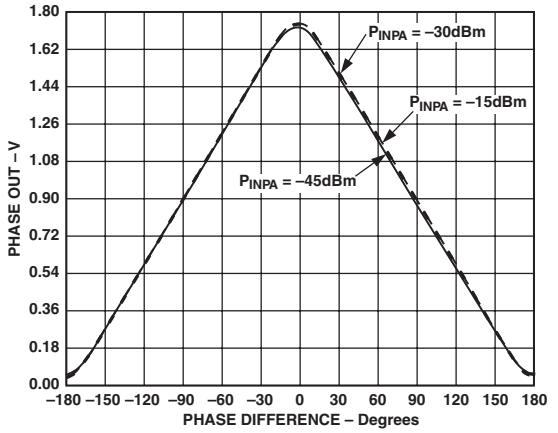
TPC 38. VPHS Output Response to 4° Step with Nominal Phase Shift of 90°, Input Levels –30 dBm, Frequency 1900 MHz, 25°C, 1 pF Filter Capacitor



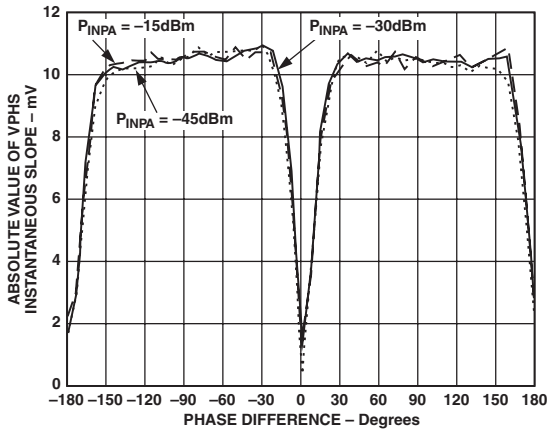
TPC 41. VPHS Output Noise Spectral Density vs. Frequency, P_{INPA} = –30 dBm, P_{INPB} = –10 dBm, –30 dBm, –50 dBm, and 90° Input Phase Difference



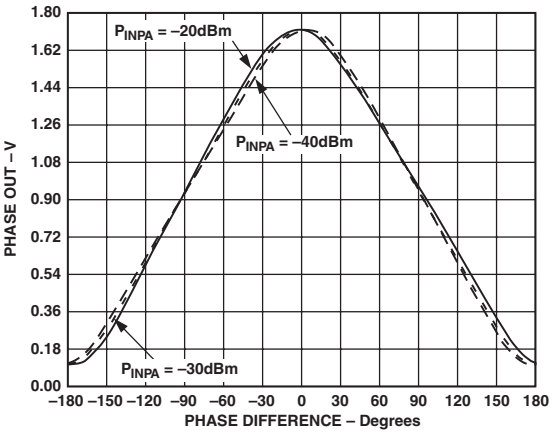
TPC 39. VPHS Output Response to 4° Step with Nominal Phase Shift of 90°, Input Levels P_{INPA} = P_{INPB} = –30 dBm, Supply 5 V, Frequency 1900 MHz, 25°C, with 100 pF Filter Capacitor



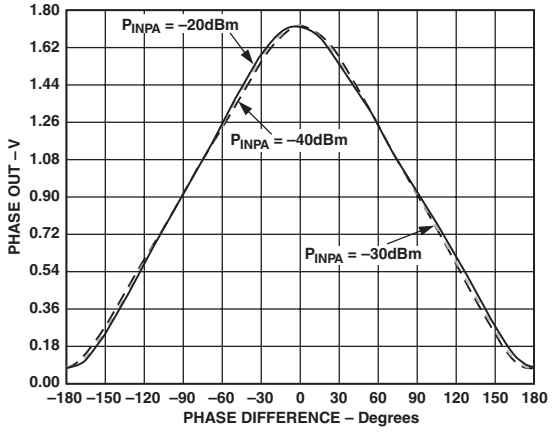
TPC 42. Phase Output vs. Input Phase Difference, P_{INPA} = P_{INPB}, P_{INPA} = P_{INPB} + 15 dB, P_{INPA} = P_{INPB} – 15 dB, Frequency 900 MHz



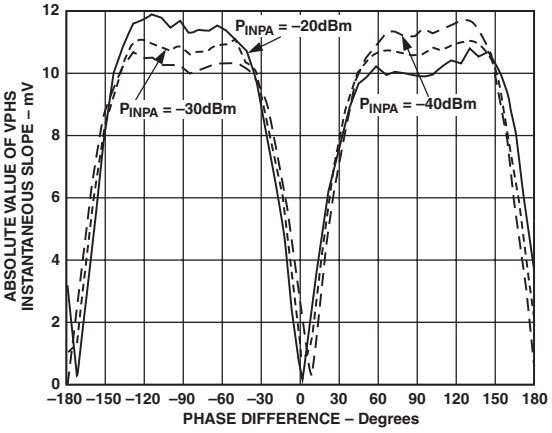
TPC 43. Phase Output Instantaneous Slope, $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB}$ + 15 dB, $P_{INPA} = P_{INPB}$ – 15 dB, Frequency 900 MHz



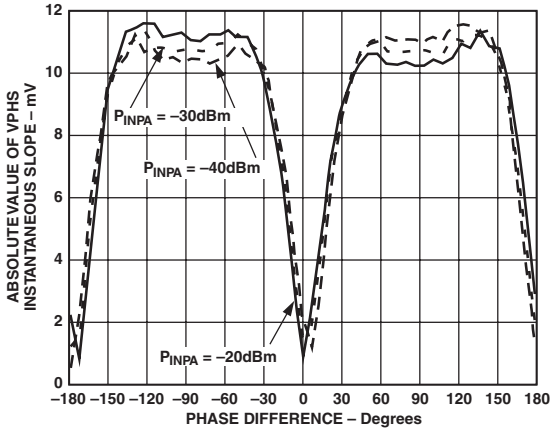
TPC 46. Phase Output vs. Input Phase Difference, $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB}$ + 10 dB, $P_{INPA} = P_{INPB}$ – 10 dB, Frequency 2200 MHz



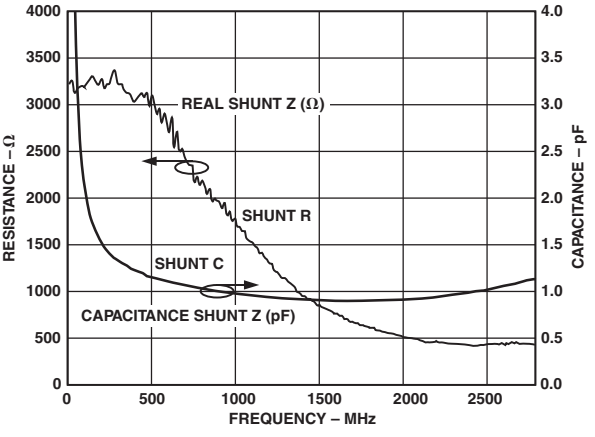
TPC 44. Phase Output vs. Input Phase Difference, $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB}$ + 10 dB, $P_{INPA} = P_{INPB}$ – 10 dB, Frequency 1900 MHz, Supply 5 V



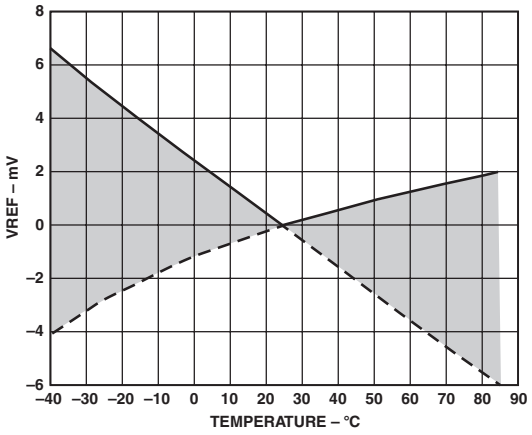
TPC 47. Phase Output Instantaneous Slope, $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB}$ + 10 dB, $P_{INPA} = P_{INPB}$ – 10 dB, Frequency 2200 MHz



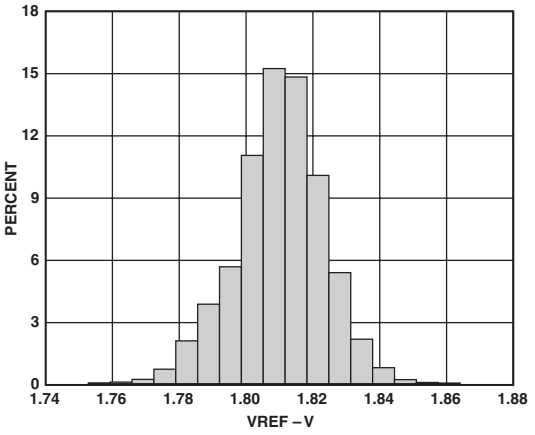
TPC 45. Phase Output Instantaneous Slope, $P_{INPA} = P_{INPB}$, $P_{INPA} = P_{INPB}$ + 10 dB, $P_{INPA} = P_{INPB}$ – 10 dB, Frequency 1900 MHz, Supply 5 V



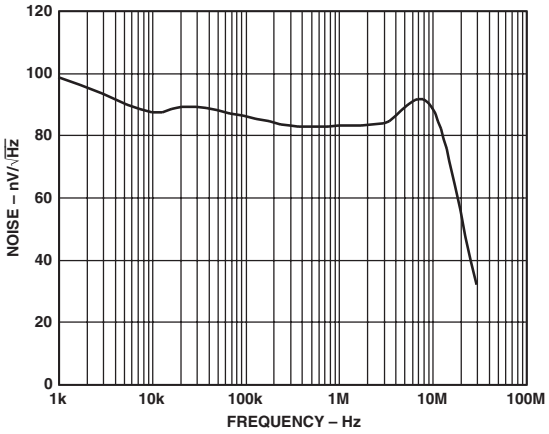
TPC 48. Input Impedance, Modeled as Shunt R in Parallel with Shunt C



TPC 49. Change in VREF vs. Temperature, Three Sigma to Either Side of Mean



TPC 51. VREF Distribution, 17,000 Units



TPC 50. VREF Output Noise Spectral Density vs. Frequency

GENERAL DESCRIPTION AND THEORY

The AD8302 measures the magnitude ratio, defined here as gain, and phase difference between two signals. A pair of matched logarithmic amplifiers provide the measurement, and their hard-limited outputs drive the phase detector.

Basic Theory

Logarithmic amplifiers (log amps) provide a logarithmic compression function that converts a large range of input signal levels to a compact decibel-scaled output. The general mathematical form is:

$$V_{OUT} = V_{SLP} \log\left(V_{IN} / V_Z\right) \quad (1)$$

where V_{IN} is the input voltage, V_Z is called the intercept (voltage), and V_{SLP} is called the slope (voltage). It is assumed throughout that log(x) represents the log10(x) function. V_{SLP} is thus the volts/decade, and since a decade of voltage corresponds to 20 dB, V_{SLP}/20 is the volts/dB. V_Z is the value of input signal that results in an output of zero and need not correspond to a physically realizable part of the log amp signal range. While the slope is fundamentally a characteristic of the log amp, the intercept is a function of the input waveform as well.[1] Furthermore, the intercept is typically more sensitive to temperature and frequency than the slope. When single log amps are used for power measurement, this variability introduces errors into the absolute accuracy of the measurement since the intercept represents a reference level.

The AD8302 takes the difference in the output of two identical log amps, each driven by signals of similar waveforms but at different levels. Since subtraction in the logarithmic domain corresponds to a ratio in the linear domain, the resulting output becomes:

$$V_{MAG} = V_{SLP} \log\left(V_{INA} / V_{INB}\right) \quad (2)$$

where V_{INA} and V_{INB} are the input voltages, V_{MAG} is the output corresponding to the magnitude of the signal level difference, and V_{SLP} is the slope. Note that the intercept, V_Z, has dropped out. Unlike the measurement of power, when measuring a dimensionless quantity such as relative signal level, no independent reference or intercept need be invoked. In essence, one signal serves as the intercept for the other. Variations in intercept due to frequency, process, temperature, and supply voltage affect both channels identically and hence do not affect the difference. This technique depends on the two log amps being well matched in slope and intercept to ensure cancellation. This is the case for an integrated pair of log amps. Note that if the two signals have different waveforms (e.g., different peak-to-average ratios) or different frequencies, an intercept difference may appear, introducing a systematic offset.

The log amp structure consists of a cascade of linear/limiting gain stages with demodulating detectors. Further details about the structure and function of log amps can be found in data sheets for other log amps produced by Analog Devices.[2] The output of the final stage of a log amp is a fully limited signal over most of the input dynamic range. The limited outputs from both log amps drive an exclusive-OR style digital phase detector. Operating strictly on the relative zero-crossings of the limited signals, the extracted phase difference is independent of the original input signal levels. The phase output has the general form:

$$V_{PHS} = V_\Phi \left[\Phi\left(V_{INA}\right) - \Phi\left(V_{INB}\right)\right] \quad (3)$$

where V_Φ is the phase slope in mV/degree and Φ is each signal's relative phase in degrees.

Structure

The general form of the AD8302 is shown in Figure 2. The major blocks consist of two demodulating log amps, a phase detector, output amplifiers, a biasing cell, and an output reference voltage buffer. The log amps and phase detector process the high frequency signals and deliver the gain and phase information in current form to the output amplifiers. The output amplifiers determine the final gain and phase scaling. External filter capacitors set the averaging time constants for the respective outputs. The reference buffer provides a 1.80 V reference voltage that tracks the internal scaling constants.

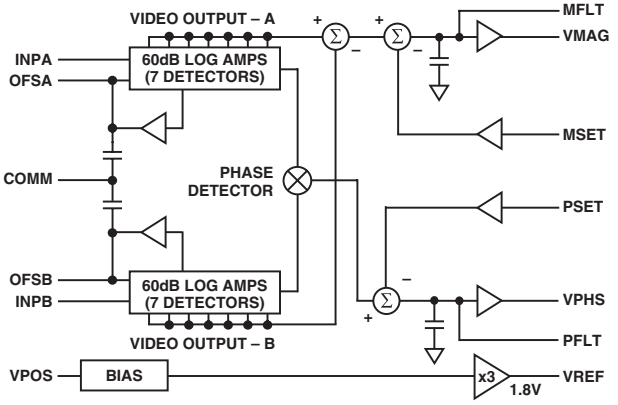


Figure 2. General Structure

Each log amp consists of a cascade of six 10 dB gain stages with seven associated detectors. The individual gain stages have 3 dB bandwidths in excess of 5 GHz. The signal path is fully differential to minimize the effect of common-mode signals and noise. Since there is a total of 60 dB of cascaded gain, slight dc offsets can cause limiting of the latter stages, which may cause measurement errors for small signals. This is corrected by a feedback loop. The nominal high-pass corner frequency, f_{HP}, of this loop is set internally at 200 MHz but can be lowered by adding external capacitance to the OFSA and OFSB pins. Signals at frequencies well below the high-pass corner are indistinguishable from dc offsets and are also nulled. The difference in the log amp outputs is performed in the current domain, yielding by analogy to Equation 2:

$$I_{LA} = I_{SLP} \log\left(V_{INA} / V_{INB}\right) \quad (4)$$

where I_{LA} and I_{SLP} are the output current difference and the characteristic slope (current) of the log amps, respectively. The slope is derived from an accurate reference designed to be insensitive to temperature and supply voltage.

The phase detector uses a fully symmetric structure with respect to its two inputs to maintain balanced delays along both signal paths. Fully differential signaling again minimizes the sensitivity to common-mode perturbations. The current-mode equivalent to Equation 3 is:

$$I_{PD} = I_\Phi \left[\Phi\left(V_{INA}\right) - \Phi\left(V_{INB}\right) - 90^\circ\right] \quad (5)$$

where I_{PD} and I_Φ are the output current and characteristic slope associated with the phase detector, respectively. The slope is derived from the same reference as the log amp slope.

NOTES

[1]See the data sheet for the AD640 for a description of the effect of waveform on the intercept of log amps.

[2]For example, see the data sheet for the AD8307.

Note that by convention, the phase difference is taken in the range from –180° to +180°. Since this style of phase detector does not distinguish between ±90°, it is considered to have an unambiguous 180° phase difference range that can be either 0° to +180° centered at +90° or 0° to –180° centered at –90°.

The basic structure of both output interfaces is shown in Figure 3. It accepts a setpoint input and includes an internal integrating/averaging capacitor and a buffer amplifier with gain K. External access to these setpoints provides for several modes of operation and enables flexible tailoring of the gain and phase transfer characteristics. The setpoint interface block, characterized by a transresistance R_F, generates a current proportional to the voltage presented to its input pin, MSET or PSET. A precise offset voltage of 900 mV is introduced internally to establish the center-point (V_{CP}) for the gain and phase functions, i.e., the setpoint voltage that corresponds to a gain of 0 dB and a phase difference of 90°. This setpoint current is subtracted from the signal current, I_{IN}, coming from the log amps in the gain channel or from the phase detector in the phase channel. The resulting difference is integrated on the averaging capacitors at either pin MFLT or PFLT and then buffered by the output amplifier to the respective output pins, VMAG and VPHS. With this open-loop arrangement, the output voltage is a simple integration of the difference between the measured gain/phase and the desired setpoint:

$$V_{OUT} = R_F\left(I_{IN} - I_{FB}\right) / \left(sT\right) \tag{6}$$

where I_{FB} is the feedback current equal to $(V_{SET} - V_{CP})/R_F$, V_{SET} is the setpoint input, and T is the integration time constant equal to $R_F C_{AVE}/K$, where C_{AVE} is the parallel combination of the internal 1.5 pF and the external capacitor C_{FLT}.

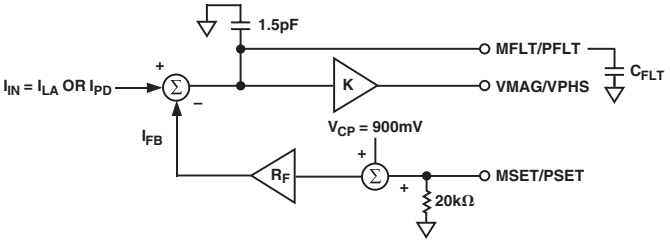


Figure 3. Simplified Block Diagram of the Output Interface

BASIC CONNECTIONS

Measurement Mode

The basic function of the AD8302 is the direct measurement of gain and phase. When the output pins, VMAG and VPHS, are connected directly to the feedback setpoint input pins, MSET and PSET, the default slopes and center points are invoked. This basic connection shown in Figure 4 is termed the measurement mode. The current from the setpoint interface is forced by the integrator to be equal to the signal currents coming from the log amps and phase detector. The closed loop transfer function is thus given by:

$$V_{OUT} = \left(I_{IN} R_F + V_{CP}\right) / \left(1 + sT\right) \tag{7}$$

The time constant T represents the single-pole response to the envelope of the dB-scaled gain and the degree-scaled phase functions. A small internal capacitor sets the maximum envelope bandwidth to approximately 30 MHz. If no external C_{FLT} is used, the AD8302 can follow the gain and phase envelopes within this bandwidth. If longer averaging is desired, C_{FLT} can be added as necessary according to T (ns) = 3.3 × C_{AVE} (pF). For best transient response with minimal overshoot, it is recommended that 1 pF minimum value external capacitors be added to the MFLT and PFLT pins.

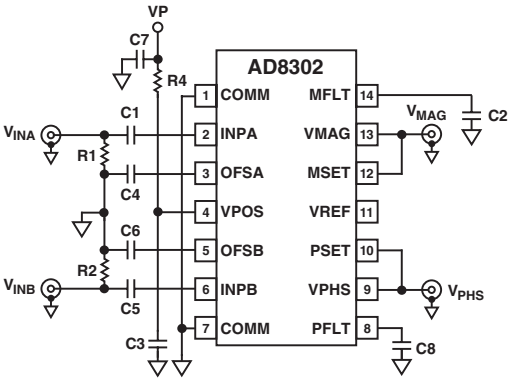


Figure 4. Basic Connections in Measurement Mode with 30 mV/dB and 10 mV/Degree Scaling

In the low frequency limit, the gain and phase transfer functions given in Equations 4 and 5 become:

$$V_{MAG} = R_F I_{SLP} \log\left(V_{INA} / V_{INB}\right) + V_{CP} \text{ or} \tag{8a}$$

$$V_{MAG} = \left(R_F I_{SLP} / 20\right)\left(P_{INA} - P_{INB}\right) + V_{CP} \tag{8b}$$

$$V_{PHS} = -R_F I_\Phi \left(\left|\Phi\left(V_{INA}\right) - \Phi\left(V_{INB}\right)\right| - 90°\right) + V_{CP} \tag{9}$$

which are illustrated in Figure 5. In Equation 8b, P_{INA} and P_{INB} are the power in dBm equivalent to V_{INA} and V_{INB} at a specified reference impedance. For the gain function, the slope represented by $R_F I_{SLP}$ is 600 mV/decade or, dividing by 20 dB/decade, 30 mV/dB. With a center point of 900 mV for 0 dB gain, a range of –30 dB to +30 dB covers the full-scale swing from 0 V to 1.8 V. For the phase function, the slope represented by $R_F I_\Phi$ is 10 mV/degree. With a center point of 900 mV for 90°, a range of 0° to 180° covers the full-scale swing from 1.8 V to 0 V. The range of 0° to –180° covers the same full-scale swing but with the opposite slope.

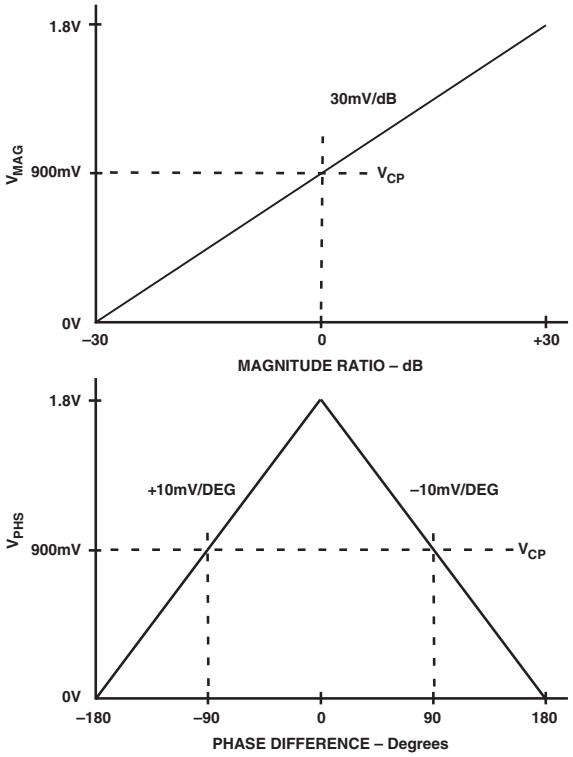


Figure 5. Idealized Transfer Characteristics for the Gain and Phase Measurement Mode

Interfacing to the Input Channels

The single-ended input interfaces for both channels are identical. Each consists of a driving pin, INPA and INPB, and an ac-grounding pin, OFSA and OFSB. All four pins are internally dc-biased at about 100 mV from the positive supply and should be externally ac-coupled to the input signals and to ground. For the signal pins, the coupling capacitor should offer negligible impedance at the signal frequency. For the grounding pins, the coupling capacitor has two functions: It provides ac grounding and sets the high-pass corner frequency for the internal offset compensation loop. There is an internal 10 pF capacitor to ground that sets the maximum corner to approximately 200 MHz. The corner can be lowered according the formula f_{HP} (MHz) = $2/C_C$(nF), where C_C is the total capacitance from OFSA or OFSB to ground, including the internal 10 pF.

The input impedance to INPA and INPB is a function of frequency, the offset compensation capacitor, and package parasitics. At moderate frequencies above f_{HP}, the input network can be approximated by a shunt 3 kΩ resistor in parallel with a 2 pF capacitor. At higher frequencies, the shunt resistance decreases to approximately 500 Ω. The Smith Chart in Figure 6 shows the input impedance over the frequency range 100 MHz to 3 GHz.

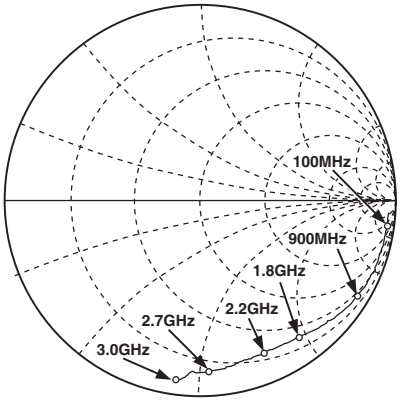


Figure 6. Smith Chart Showing the Input Impedance of a Single Channel from 100 MHz to 3 GHz

A broadband resistive termination on the signal side of the coupling capacitors can be used to match to a given source impedance. The value of the termination resistor, R_T, is determined by:

$$R_T = R_{IN}R_S / \left(R_{IN} - R_S\right) \quad (10)$$

where R_{IN} is the input resistance and R_S the source impedance. At higher frequencies, a reactive, narrow-band match might be desirable to tune out the reactive portion of the input impedance. An important attribute of the two-log-amp architecture is that if both channels are at the same frequency and have the same input network, then impedance mismatches and reflection losses become essentially common-mode and hence do not impact the relative gain and phase measurement. However, mismatches in these external components can result in measurement errors.

Dynamic Range

The maximum measurement range for the gain subsystem is limited to a total of 60 dB distributed from –30 dB to +30 dB. This means that both gain and attenuation can be measured. The limits are determined by the minimum and maximum levels that each individual log amp can detect. In the AD8302, each log amp can detect inputs ranging from –73 dBV [(223 μV, –60 dBm re: 50 Ω to –13 dBV (223 mV, 0 dBm re: 50 Ω)]. Note that log amps respond to voltages and not power. An equivalent power can be inferred given an impedance level, e.g., to convert from dBV to dBm in a 50 Ω system, simply add 13 dB. To cover the entire range, it is necessary to apply a reference level to one log amp that corresponds precisely to its midrange. In the AD8302, this level is at –43 dBV, which corresponds to –30 dBm in a 50 Ω environment. The other channel can now sweep from its low end, 30 dB below midrange, to its high end, 30 dB above midrange. If the reference is displaced from midrange, some measurement range will be lost at the extremes. This can occur either if the log amps run out of range or if the rails at ground or 1.8 V are reached. Figure 7 illustrates the effect of the reference channel level placement. If the reference is chosen lower than midrange by 10 dB, then the lower limit will be at –20 dB rather than –30 dB. If the reference chosen is higher by 10 dB, the upper limit will be 20 dB rather than 30 dB.

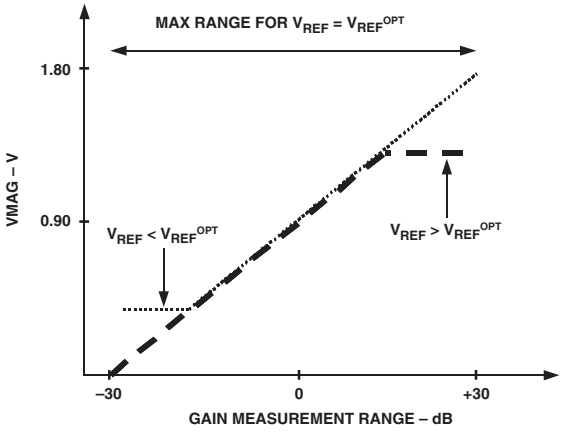


Figure 7. The Effect of Offsetting the Reference Level Is to Reduce the Maximum Dynamic Range

The phase measurement range is of 0° to 180°. For phase differences of 0° to –180°, the transfer characteristics are mirrored as shown in Figure 5, with a slope of the opposite sign. The phase detector responds to the relative position of the zero crossings between the two input channels. At higher frequencies, the finite rise and fall times of the amplitude limited inputs create an ambiguous situation that leads to inaccessible dead zones at the 0° and 180° limits. For maximum phase difference coverage, the reference phase difference should be set to 90°.

Cross Modulation of Magnitude and Phase

At high frequencies, unintentional cross coupling between signals in Channels A and B inevitably occurs due to on-chip and board-level parasitics. When the two signals presented to the AD8302 inputs are at very different levels, the cross coupling introduces cross modulation of the phase and magnitude responses. If the two signals are held at the same relative levels and the phase between them is modulated then only the phase output should respond. Due to phase-to-amplitude cross modulation, the magnitude output shows a residual response. A similar effect occurs when the relative phase is held constant while the magnitude difference is modulated, i.e., an expected magnitude response and a residual phase response are observed due to amplitude-to-phase cross modulation. The point where these effects are noticeable depends on the signal frequency and the magnitude of the difference. Typically, for differences <20 dB, the effects of cross modulation are negligible at 900 MHz.

Modifying the Slope and Center Point

The default slope and center point values can be modified with the addition of external resistors. Since the output interface blocks are generalized for both magnitude and phase functions, the scaling modification techniques are equally valid for both outputs. Figure 8 demonstrates how a simple voltage divider from the VMAG and VPHS pins to the MSET and PSET pins can be used to modify the slope. The increase in slope is given by 1 + R1/(R2‖20 kΩ). Note that it may be necessary to account for the MSET and PSET input impedance of 20 kΩ which has a ±20% manufacturing tolerance. As is generally true in such feedback systems, envelope bandwidth is decreased and the output noise transferred from the input is increased by the same factor. For example, by selecting R1 and R2 to be 10 kΩ and 20 kΩ, respectively, gain slope increases from the nominal 30 mV/dB by a factor of 2 to 60 mV/dB. The range is reduced by a factor of 2 and the new center point is at –15 dB, i.e., the range now extends from –30 dB, corresponding to V_{MAG} = 0 V, to 0 dB, corresponding to V_{MAG} = 1.8 V.

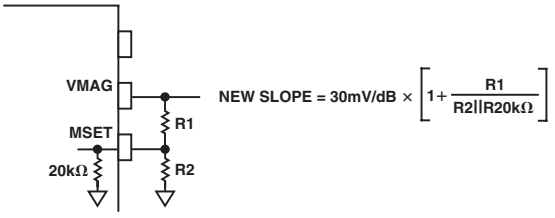


Figure 8. Increasing the Slope Requires the Inclusion of a Voltage Divider

Repositioning the center point back to its original value of 0 dB simply requires that an appropriate voltage be applied to the grounded side of the lower resistor in the voltage divider. This voltage may be provided externally or derived from the internal reference voltage on pin VREF. For the specific choice of R2 = 20 kΩ, the center point is easily readjusted to 0 dB by connecting the VREF pin directly to the lower pin of R2 as shown in Figure 9. The increase in slope is now simplified to 1 + R1/10 kΩ. Since this 1.80 V reference voltage is derived from the same band gap reference that determines the nominal center point, their tracking with temperature, supply, and part-to-part variations should be better in comparison to a fixed external voltage. If the center point is shifted to 0 dB in the previous example where the slope was doubled, then the range spans from –15 dB at V_{MAG} = 0 V to 15 dB at V_{MAG} = 1.8 V.

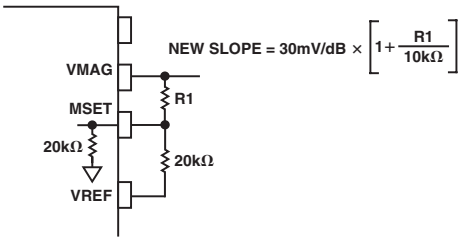


Figure 9. The Center Point Is Repositioned with the Help of the Internal Reference Voltage of 1.80 V

Comparator and Controller Modes

The AD8302 can also operate in a comparator mode if used in the arrangement shown in Figure 10 where the DUT is the element to be evaluated. The VMAG and VPHS pins are no longer connected to MSET and PSET. The trip-point thresholds for the gain and phase difference comparison are determined by the voltages applied to pins MSET and PSET according to:

$$V_{MSET}(V) = 30\ mV/dB \times Gain^{SP}(dB) + 900\ mV \tag{11}$$

$$V_{PSET}(V) = -10\ mV/^\circ \times \left(|\ Phase^{SP}(^\circ)\ | - 90^\circ\right) + 900\ mV \tag{12}$$

where $Gain^{SP}$ (dB) and $Phase^{SP}$ $(^\circ)$ are the desired gain and phase thresholds. If the actual gain and phase between the two input channels differ from these thresholds, the V_{MAG} and V_{PHS} outputs toggle like comparators, i.e.,

$$V_{MAG} = \begin{cases} 1.8\,V \text{ if } Gain > Gain^{SP} \\ 0\,V \text{ if } Gain < Gain^{SP} \end{cases} \tag{13}$$

$$V_{PHS} = \begin{cases} 1.8\,V \text{ if } Phase > Phase^{SP} \\ 0\,V \text{ if } Phase < Phase^{SP} \end{cases} \tag{14}$$

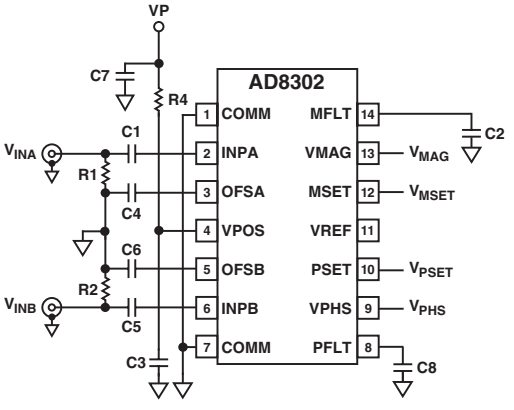


Figure 10. Disconnecting the Feedback to the Setpoint Controls, the AD8302 Operates in Comparator Mode

The comparator mode can be turned into a controller mode by closing the loop around the VMAG and VPHS outputs. Figure 11 illustrates a closed loop controller that stabilizes the gain and phase of a DUT with gain and phase adjustment elements. If VMAG and VPHS are properly conditioned to drive gain and phase adjustment blocks preceding the DUT, the actual gain and phase of the DUT will be forced toward the prescribed setpoint gain and phase given in Equations 11 and 12. These are essentially AGC and APC loops. Note that as with all control loops of this kind, loop dynamics and appropriate interfaces all must be considered in more detail.

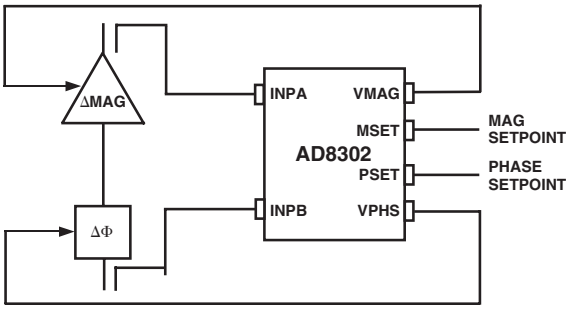


Figure 11. By Applying Overall Feedback to a DUT Via External Gain and Phase Adjusters, the AD8302 Acts as a Controller

APPLICATIONS

Measuring Amplifier Gain and Compression

The most fundamental application of AD8302 is the monitoring of the gain and phase response of a functional circuit block such as an amplifier or a mixer. As illustrated in Figure 12, directional couplers, DC_B and DC_A, sample the input and output signals of the "Black Box" DUT. The attenuators ensure that the signal levels presented to the AD8302 fall within its dynamic range. From the discussion in the Dynamic Range section, the optimal choice places both channels at P_{OPT} = –30 dBm referenced to 50 Ω, which corresponds to –43 dBV. To achieve this, the combination of coupling factor and attenuation are given by:

$$C_B + L_B = P_{IN} - P_{OPT} \quad (15)$$

$$C_A + L_A = P_{IN} + GAIN_{NOM} - P_{OPT} \quad (16)$$

where C_B and C_A are the coupling coefficients, L_B and L_A are the attenuation factors, and $GAIN_{NOM}$ is the nominal DUT gain. If identical couplers are used for both ports, then the difference in the two attenuators compensates for the nominal DUT gain. When the actual gain is nominal, the VMAG output is 900 mV, corresponding to 0 dB. Variations from nominal gain appear as a deviation from 900 mV or 0 dB with a 30 mV/dB scaling. Depending on the nominal insertion phase associated with DUT, the phase measurement may require a fixed phase shift in series with one of the channels to bring the nominal phase difference presented to the AD8302 near the optimal 90° point.

When the insertion phase is nominal, the VPHS output is 900 mV. Deviations from the nominal are reported with a 10 mV/degree scaling. Table I gives suggested component values for the measurement of an amplifier with a nominal gain of 10 dB and an input power of –10 dBm.

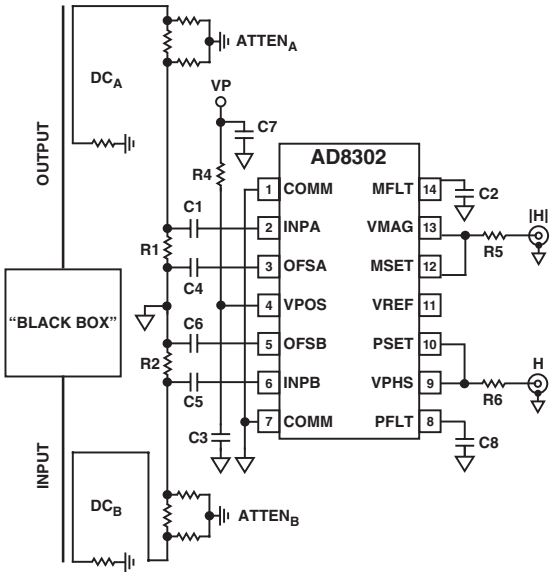


Figure 12. Using the AD8302 to Measure the Gain and Insertion Phase of an Amplifier or Mixer

Table I. Component Values for Measuring a 10 dB Amplifier with an Input Power of –10 dBm

Component	Value	Quantity
R1, R2	52.3 Ω	2
R5, R6	100 Ω	2
C1, C4, C5, C6	0.001 μF	4
C2, C8	Open	
C3	100 pF	1
C7	0.1 μF	1
AttenA	10 dB (See Text)	1
AttenB	1 dB (See Text)	1
DC_A, DC_B	20 dB	2

The gain measurement application can also monitor gain and phase distortion in the form of AM-AM (gain compression) and AM-PM conversion. In this case, the nominal gain and phase corresponds to those at low input signal levels. As the input level is increased, output compression and excess phase shifts are measured as deviations from the low level case. Note that the signal levels over which the input is swept must remain within the dynamic range of the AD8302 for proper operation.

Reflectometer

The AD8302 can be configured to measure the magnitude ratio and phase difference of signals that are incident on and reflected from a load. The vector reflection coefficient, Γ, is defined as,

$$\Gamma = \textit{Reflected Voltage} / \textit{Incident Voltage} = \left(Z_L - Z_O\right) / \left(Z_L + Z_O\right) \quad (17)$$

where Z_L is the complex load impedance and Z_O is the characteristic system impedance.

The measured reflection coefficient can be used to calculate the level of impedance mismatch or standing wave ratio (SWR) of a particular load condition. This proves particularly useful in diagnosing varying load impedances such as antennas that can degrade performance and even cause physical damage. The vector reflectometer arrangement given in Figure 13 consists of a pair of directional couplers that sample the incident and reflected signals. The attenuators reposition the two signal levels within the dynamic range of the AD8302. In analogy to Equations 15 and 16, the attenuation factors and coupling coefficients are given by:

$$C_B + L_B = P_{IN} - P_{OPT} \quad (18)$$

$$C_A + L_A = P_{IN} + \Gamma_{NOM} - P_{OPT} \quad (19)$$

where Γ_{NOM} is the nominal reflection coefficient in dB and is negative for passive loads. Consider the case where the incident signal is 10 dBm and the nominal reflection coefficient is –19 dB. As shown in Figure 13, using 20 dB couplers on both sides and –30 dBm for P_{OPT}, the attenuators for Channel A and B paths are 1 dB and 20 dB, respectively. The magnitude and phase of the reflection coefficient are available at the VMAG and VPHS pins scaled to 30 mV/dB and 10 mV/degree. When Γ is –19 dB, the VMAG output is 900 mV.

The measurement accuracy can be compromised if board level details are not addressed. Minimize the physical distance between the series connected couplers since the extra path length adds phase error to Γ. Keep the paths from the couplers to the AD8302 as well matched as possible since any differences introduce measurement errors. The finite directivity, D, of the couplers sets the minimum detectable reflection coefficient, i.e., $|\Gamma_{MIN}(dB)| < |D(dB)|$.

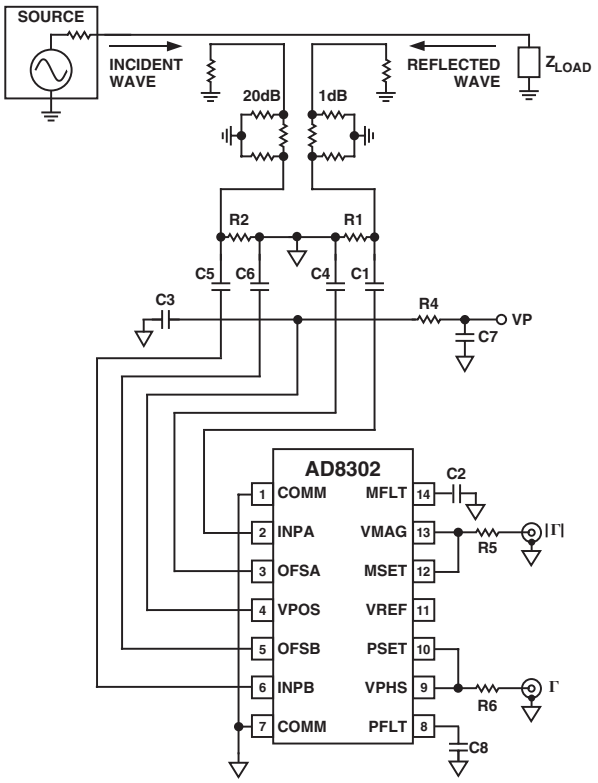


Figure 13. Using the AD8302 to Measure the Vector Reflection Coefficient Off an Arbitrary Load

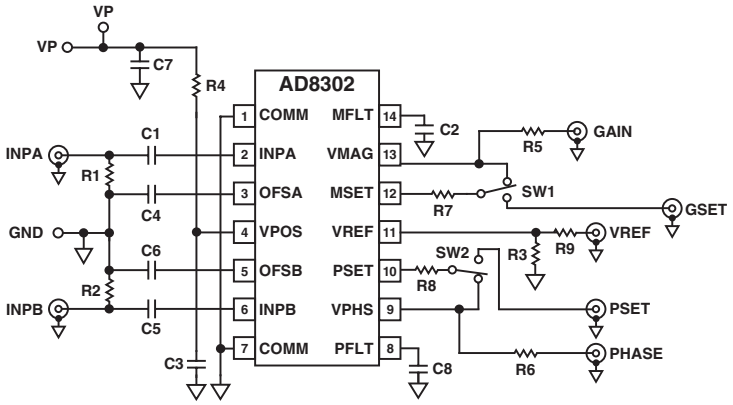


Figure 14. Evaluation Board Schematic

Table II. P1 Pin Allocations

1	Common
2	VPOS
3	Common

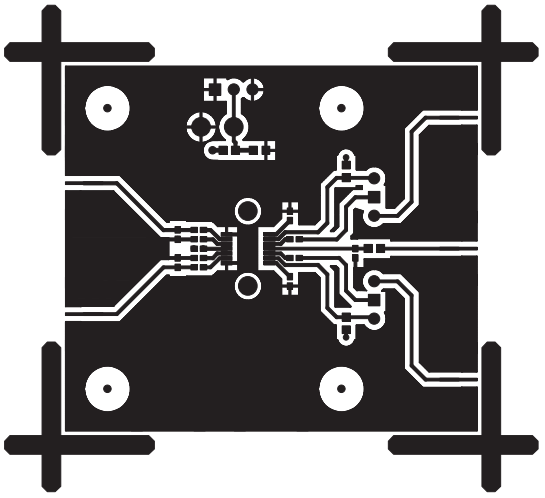


Figure 15a. Component Side Metal of Evaluation Board

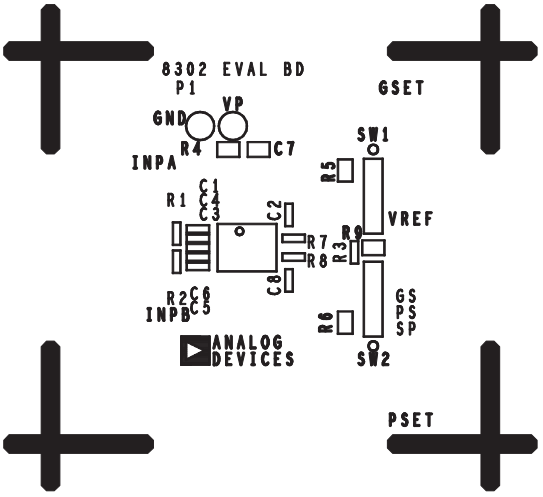


Figure 15b. Component Side Silkscreen of Evaluation Board

Table III. Evaluation Board Configuration Options

Component	Function	Default Condition
P1	Power Supply and Ground Connector: Pin 2 VPOS and Pins 1 and 3 Ground.	Not Applicable
R1, R2	Input Termination. Provide termination for input sources.	R1 = R2 = 52.3 Ω (Size 0402)
R3	VREF Output Load. This load is optional and is meant to allow the user to simulate their circuit loading of the device.	R3 = 1 kΩ (Size 0603)
R5, R6, R9	Snubbing Resistor	R5 = R6 = 0 Ω (Size 0603) R9 = 0 Ω (Size 0603)
C3, C7, R4	Supply Decoupling	C3 = 100 pF (Size 0603) C7 = 0.1 µF (Size 0603) R4 = 0 Ω (Size 0603)
C1, C5	Input AC-Coupling Capacitors	C1 = C5 = 1 nF (Size 0603)
C2, C8	Video Filtering. C2 and C8 limit the video bandwidth of the gain and phase output respectively.	C2 = C8 = Open (Size 0603)
C4, C6	Offset Feedback. These set the high-pass corner of the offset cancellation loop and thus with the input ac-coupling capacitors the minimum operating frequency.	C4 = C6 = 1 nF (Size 0603)
SW1	GSET Signal Source. When SW1 is in the position shown, the device is in gain measure mode; when switched, it operates in comparator mode and a signal must be applied to GSET.	SW1 = Installed
SW2	PSET Signal Source. When SW2 is in the position shown, the device is in phase measure mode; when switched, it operates in comparator mode and a signal must be applied to PSET.	SW1 = Installed

CHARACTERIZATION SETUPS AND METHODS

The general hardware configuration used for most of the AD8302 characterization is shown in Figure 16. The characterization board is similar to the Customer Evaluation Board. Two reference-locked R and S SMT03 signal generators are used as the inputs to INPA and INPB, while the gain and phase outputs are monitored using both a TDS 744A oscilloscope with 10× high impedance probes and Agilent 34401A multimeters.

Gain

The basic technique used to evaluate the static gain (VMAG) performance was to set one source to a fixed level and sweep the amplitude of the other source, while measuring the VMAG output with the DMM. In practice, the two sources were run at 100 kHz frequency offset and average output measured with the DMM to alleviate errors that might be induced by gain/phase modulation due to phase jitter between the two sources.

The errors stated are the difference between a best fit line calculated by a linear regression and the actual measured data divided by the slope of the line to give an error in V/dB. The referred to 25°C error uses this same method while always using the slope and intercept calculated for that device at 25°C.

Response measurement made of the VMAG output used the configuration shown in Figure 17. The variable attenuator, Alpha AD260, is driven with a HP8112A pulse generator producing a change in RF level within 10 ns.

Noise spectral density measurements were made using a HP3589A with the inputs delivered through a Narda 4032C 90° phase splitter.

To measure the modulation of VMAG due to phase variation again the sources were run at a frequency offset, f_{OS}, effectively creating a continuous linear change in phase going through 360° once every $1/f_{OS}$ seconds. The VMAG output is then measured with a DSO. When perceivable, only at high frequencies and large input magnitude differences, the linearly ramping phase creates a near sinusoid output riding on the expected VMAG dc output level. The curves in TPC 24 show the peak-to-peak output level measured with averaging.

Phase

The majority of the VPHS output data was collected by generating phase change, again by operating the two input sources with a small frequency offset (normally 100 kHz) using the same configuration shown in Figure 16. Although this method gives excellent linear phase change, good for measurement of slope and linearity, it lacks an absolute phase reference point. In the curves showing swept phase, the phase at which the VPHS is the same as VPHS with no input signal is taken to be –90° and all other angles are references to there. Typical Performance Curves show two figures of merit; instantaneous slope and error. Instantaneous slope, as shown in TPCs 43, 44, 45, and 47, was calculated simply by taking the delta in VPHS over angular change for adjacent measurement points.

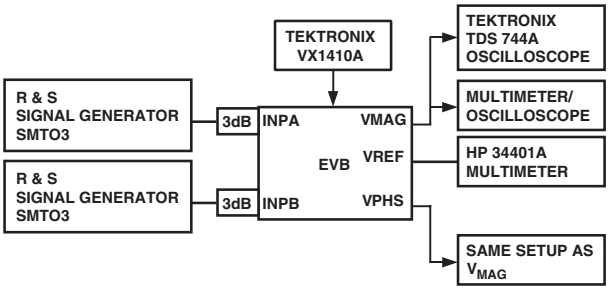


Figure 16. Primary Characterization Setup

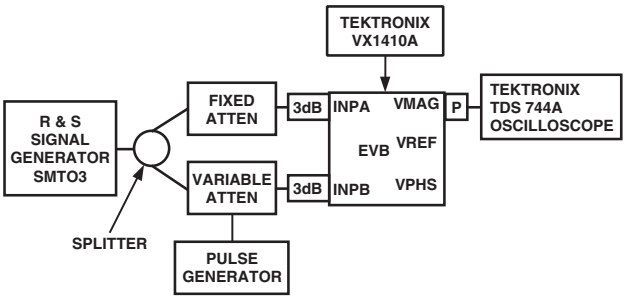
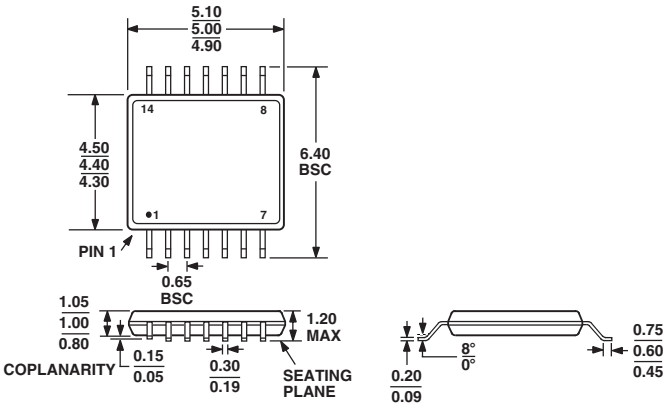


Figure 17. VMAG Dynamic Performance Measurement Setup

OUTLINE DIMENSIONS

14-Lead Thin Shrink Small Outline Package [TSSOP]
(RU-14)

Dimensions shown in millimeters



COMPLIANT TO JEDEC STANDARDS MO-153AB-1

Revision History

Location	Page
7/02—Data Sheet changed from REV. 0 to REV. A.	
TPCs 3 through 6 replaced	6

C02492–0–7/02(A)

PRINTED IN U.S.A.

E TI ADS1015 Information

Ultra-Small, Low-Power, 12-Bit Analog-to-Digital Converter with Internal Reference

Check for Samples: ADS1013 ADS1014 ADS1015

FEATURES

- **ULTRA-SMALL QFN PACKAGE: 2mm × 1,5mm × 0,4mm**
- **WIDE SUPPLY RANGE: 2.0V to 5.5V**
- **LOW CURRENT CONSUMPTION: Continuous Mode: Only 150µA Single-Shot Mode: Auto Shut-Down**
- **PROGRAMMABLE DATA RATE: 128SPS to 3.3kSPS**
- **INTERNAL LOW-DRIFT VOLTAGE REFERENCE**
- **INTERNAL OSCILLATOR**
- **INTERNAL PGA**
- **I²C™ INTERFACE: Pin-Selectable Addresses**
- **FOUR SINGLE-ENDED OR TWO DIFFERENTIAL INPUTS (ADS1015)**
- **PROGRAMMABLE COMPARATOR (ADS1014 and ADS1015)**

APPLICATIONS

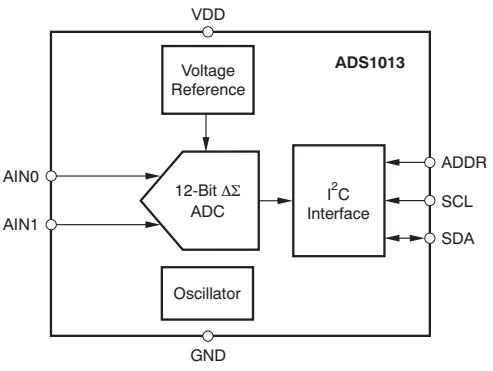
- **PORTABLE INSTRUMENTATION**
- **CONSUMER GOODS**
- **BATTERY MONITORING**
- **TEMPERATURE MEASUREMENT**
- **FACTORY AUTOMATION AND PROCESS CONTROLS**

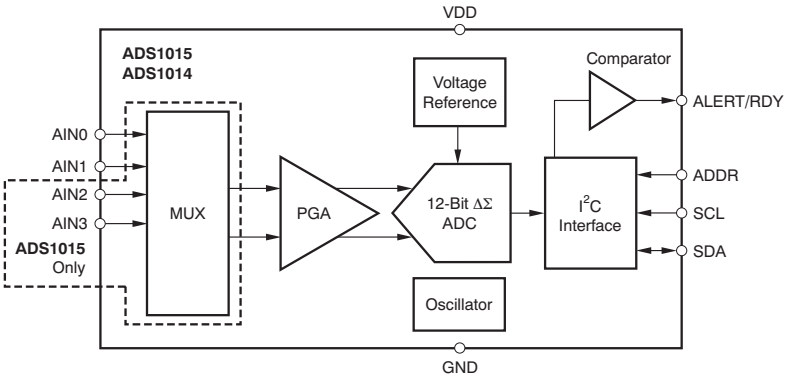
DESCRIPTION

The ADS1013, ADS1014, and ADS1015 are precision analog-to-digital converters (ADCs) with 12 bits of resolution offered in an ultra-small, leadless QFN-10 package or an MSOP-10 package. The ADS1013/4/5 are designed with precision, power, and ease of implementation in mind. The ADS1013/4/5 feature an onboard reference and oscillator. Data are transferred via an I²C-compatible serial interface; four I²C slave addresses can be selected. The ADS1013/4/5 operate from a single power supply ranging from 2.0V to 5.5V.

The ADS1013/4/5 can perform conversions at rates up to 3300 samples per second (SPS). An onboard PGA is available on the ADS1014 and ADS1015 that offers input ranges from the supply to as low as ±256mV, allowing both large and small signals to be measured with high resolution. The ADS1015 also features an input multiplexer (MUX) that provides two differential or four single-ended inputs.

The ADS1013/4/5 operate either in continuous conversion mode or a single-shot mode that automatically powers down after a conversion and greatly reduces current consumption during idle periods. The ADS1013/4/5 are specified from –40°C to +125°C.





Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

I2C is a trademark of NXP Semiconductors.
All other trademarks are the property of their respective owners.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2009, Texas Instruments Incorporated

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

ORDERING INFORMATION

For the most current package and ordering information, see the Package Option Addendum at the end of this document, or see the TI web site at www.ti.com.

ABSOLUTE MAXIMUM RATINGS(1)

	ADS1013, ADS1014, ADS1015	UNIT
VDD to GND	–0.3 to +5.5	V
Analog input current	100, momentary	mA
Analog input current	10, continuous	mA
Analog input voltage to GND	–0.3 to VDD + 0.3	V
SDA, SCL, ADDR, ALERT/RDY voltage to GND	–0.5 to +5.5	V
Maximum junction temperature	+150	°C
Storage temperature range	–60 to +150	°C

(1) Stresses above those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may affect device reliability.

PRODUCT FAMILY

DEVICE	PACKAGE DESIGNATOR MSOP/QFN	RESOLUTION (Bits)	MAXIMUM SAMPLE RATE (SPS)	COMPARATOR	PGA	INPUT CHANNELS (Differential/ Single-Ended)
ADS1113	BROI/N6J	16	860	No	No	1/1
ADS1114	BRNI/N5J	16	860	Yes	Yes	1/1
ADS1115	BOGI/N4J	16	860	Yes	Yes	2/4
ADS1013	BRMI/N9J	12	3300	No	No	1/1
ADS1014	BRQI/N8J	12	3300	Yes	Yes	1/1
ADS1015	BRPI/N7J	12	3300	Yes	Yes	2/4

Copyright © 2009, Texas Instruments Incorporated

ELECTRICAL CHARACTERISTICS

All specifications at –40°C to +125°C, VDD = 3.3V, and Full-Scale (FS) = ±2.048V, unless otherwise noted.
Typical values are at +25°C.

PARAMETER	TEST CONDITIONS	ADS1013, ADS1014, ADS1015			UNIT
		MIN	TYP	MAX	
ANALOG INPUT					
Full-scale input voltage(1)	V_{IN} = (AIN_P) – (AIN_N)		±4.096/PGA		V
Analog input voltage	AIN_P or AIN_N to GND	GND		VDD	V
Differential input impedance			See Table 2		
Common-mode input impedance	FS = ±6.144V(1)		10		MΩ
	FS = ±4.096V(1), ±2.048V		6		MΩ
	FS = ±1.024V		3		MΩ
	FS = ±0.512V, ±0.256V		100		MΩ
SYSTEM PERFORMANCE					
Resolution	No missing codes	12			Bits
Data rate (DR)			128, 250, 490, 920, 1600, 2400, 3300		SPS
Data rate variation	All data rates	–10		10	%
Output noise		See Typical Characteristics			
Integral nonlinearity	DR = 128SPS, FS = ±2.048V, best fit(2)			0.5	LSB
Offset error	FS = ±2.048V, differential inputs		0	±0.5	LSB
	FS = ±2.048V, single-ended inputs		±0.25		LSB
Offset drift	FS = ±2.048V		0.005		LSB/°C
Gain error(3)	FS = ±2.048V at 25°C		0.05	0.25	%
Gain drift(3)	FS = ±0.256V		7		ppm/°C
	FS = ±2.048V		5	40	ppm/°C
	FS = ±6.144V(1)		5		ppm/°C
PGA gain match(3)	Match between any two PGA gains		0.02	0.1	%
Gain match	Match between any two inputs		0.05	0.1	%
Offset match	Match between any two inputs		0.25		LSB
DIGITAL INPUT/OUTPUT					
Logic level					
V_{IH}		0.7VDD		5.5	V
V_{IL}		GND – 0.5		0.3VDD	V
V_{OL}	I_{OL} = 3mA	GND	0.15	0.4	V
Input leakage					
I_H	V_{IH} = 5.5V			10	µA
I_L	V_{IL} = GND	10			µA

(1) This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.
(2) 99% of full-scale.
(3) Includes all errors from onboard PGA and reference.

Copyright © 2009, Texas Instruments Incorporated

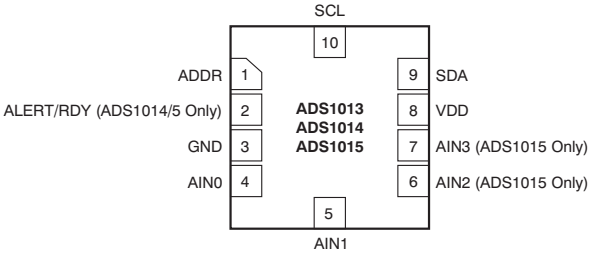
ELECTRICAL CHARACTERISTICS (continued)

All specifications at –40°C to +125°C, VDD = 3.3V, and Full-Scale (FS) = ±2.048V, unless otherwise noted.
Typical values are at +25°C.

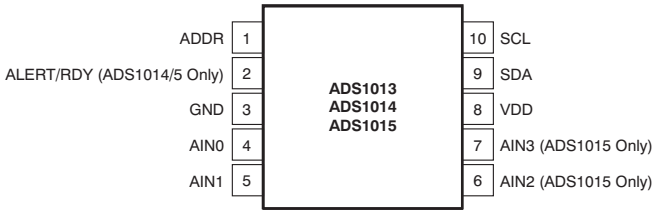
PARAMETER	TEST CONDITIONS	ADS1013, ADS1014, ADS1015			UNIT
		MIN	TYP	MAX	
POWER-SUPPLY REQUIREMENTS					
Power-supply voltage		2		5.5	V
Supply current	Power-down current at 25°C		0.5	2	µA
	Power-down current up to 125°C			5	µA
	Operating current at 25°C		150	200	µA
	Operating current up to 125°C			300	µA
Power dissipation	VDD = 5.0V		0.9		mW
	VDD = 3.3V		0.5		mW
	VDD = 2.0V		0.3		mW
TEMPERATURE					
Storage temperature		–60		+150	°C
Specified temperature		–40		+125	°C

PIN CONFIGURATIONS

RUG PACKAGE
QFN-10
(TOP VIEW)



DGS PACKAGE
MSOP-10
(TOP VIEW)



PIN DESCRIPTIONS

PIN #	DEVICE			FUNCTION	DESCRIPTION
	ADS1013	ADS1014	ADS1015		
1	ADDR	ADDR	ADDR	Digital input	I^2C slave address select
2	NC(1)	ALERT/RDY	ALERT/RDY	Digital output	Digital comparator output or conversion ready (NC for ADS1013)
3	GND	GND	GND	Supply	Ground
4	AIN0	AIN0	AIN0	Analog input	Differential channel 1: Positive input or single-ended channel 1 input
5	AIN1	AIN1	AIN1	Analog input	Differential channel 1: Negative input or single-ended channel 2 input
6	NC	NC	AIN2	Analog input	Differential channel 2: Positive input or single-ended channel 3 input (NC for ADS1013/4)
7	NC	NC	AIN3	Analog input	Differential channel 2: Negative input or single-ended channel 4 input (NC for ADS1013/4)
8	VDD	VDD	VDD	Supply	Power supply: 2.0V to 5.5V
9	SDA	SDA	SDA	Digital I/O	Serial data: Transmits and receives data
10	SCL	SCL	SCL	Digital input	Serial clock input: Clocks data on SDA

(1) NC pins may be left floating or tied to ground.

Copyright © 2009, Texas Instruments Incorporated

TIMING REQUIREMENTS

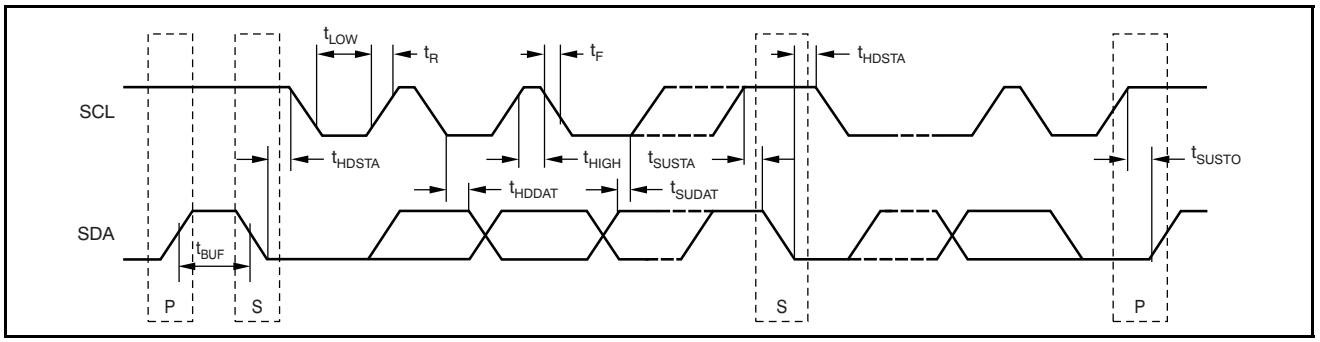


Figure 1. I²C Timing Diagram

Table 1. I²C Timing Definitions

PARAMETER		FAST MODE		HIGH-SPEED MODE		UNIT
		MIN	MAX	MIN	MAX	
SCL operating frequency	f_{SCL}	0.01	0.4	0.01	3.4	MHz
Bus free time between START and STOP condition	t_{BUF}	600		160		ns
Hold time after repeated START condition. After this period, the first clock is generated.	t_{HDSTA}	600		160		ns
Repeated START condition setup time	t_{SUSTA}	600		160		ns
Stop condition setup time	t_{SUSTO}	600		160		ns
Data hold time	t_{HDDAT}	0		0		ns
Data setup time	t_{SUDAT}	100		10		ns
SCL clock low period	t_{LOW}	1300		160		ns
SCL clock high period	t_{HIGH}	600		60		ns
Clock/data fall time	t_F		300		160	ns
Clock/data rise time	t_R		300		160	ns

Copyright © 2009, Texas Instruments Incorporated

TYPICAL CHARACTERISTICS

At T_A = +25°C and VDD = 3.3V, unless otherwise noted.

OPERATING CURRENT vs TEMPERATURE

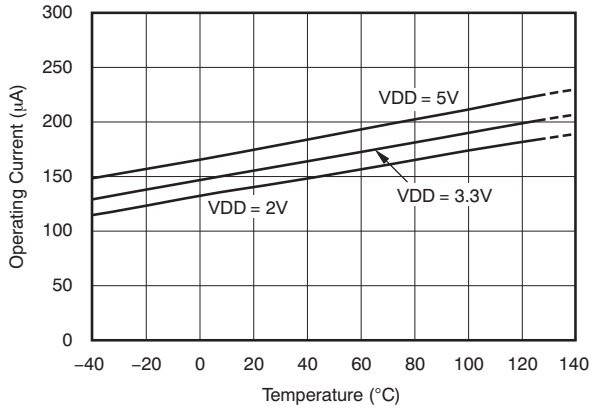


Figure 2.

SHUTDOWN CURRENT vs TEMPERATURE

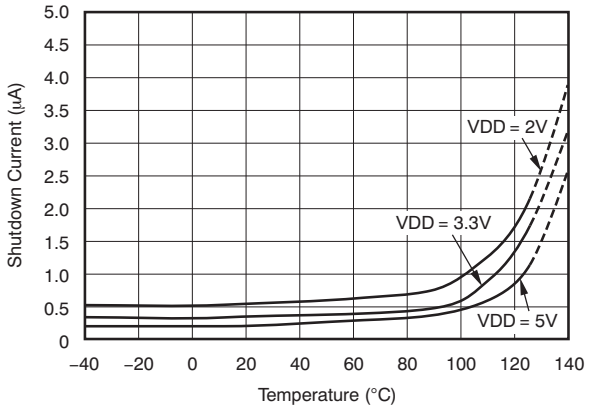


Figure 3.

SINGLE-ENDED OFFSET ERROR vs TEMPERATURE(1)

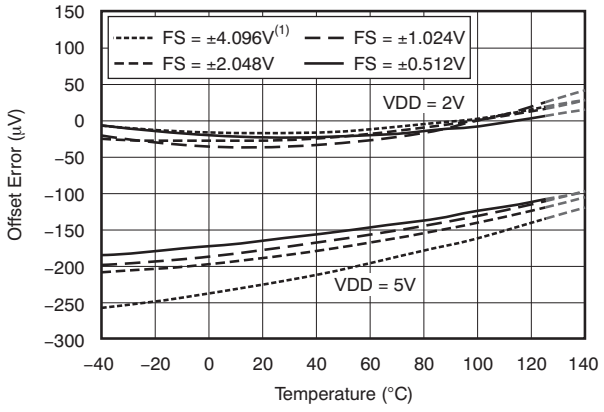


Figure 4.

DIFFERENTIAL OFFSET vs TEMPERATURE

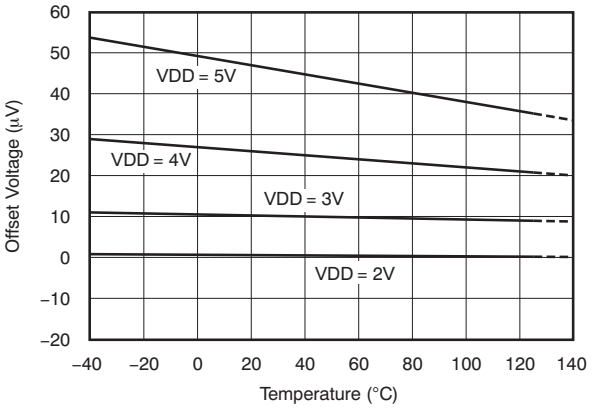


Figure 5.

GAIN ERROR vs TEMPERATURE

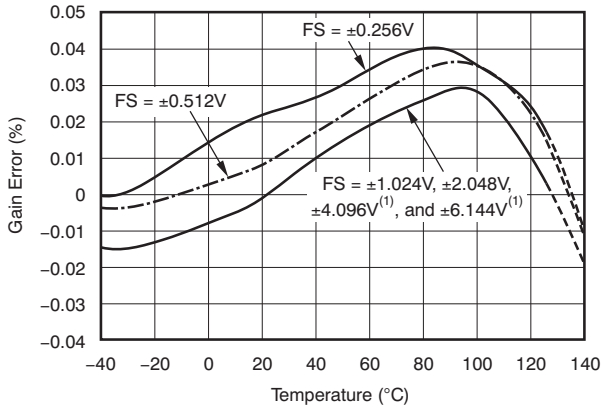


Figure 6.

NOISE PLOT

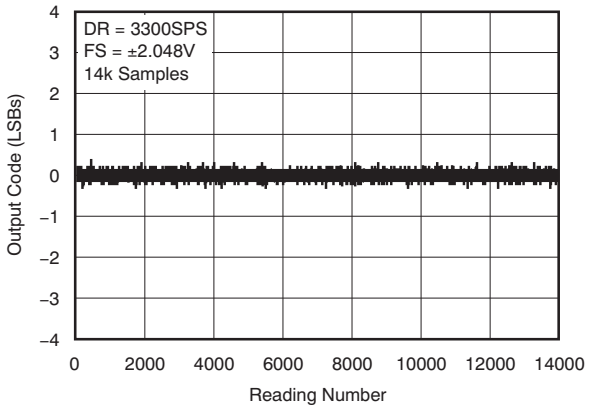


Figure 7.

(1) This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.

Copyright © 2009, Texas Instruments Incorporated

OVERVIEW

The ADS1013/4/5 are very small, low-power, 12-bit, delta-sigma (ΔΣ) analog-to-digital converters (ADCs). The ADS1013/4/5 are extremely easy to configure and design into a wide variety of applications, and allow precise measurements to be obtained with very little effort. Both experienced and novice users of data converters find designing with the ADS1013/4/5 family to be intuitive and problem-free.

The ADS1013/4/5 consist of a ΔΣ analog-to-digital (A/D) core with adjustable gain (excludes the ADS1013), an internal voltage reference, a clock oscillator, and an I^2C interface. An additional feature available on the ADS1014/5 is a programmable digital comparator that provides an alert on a dedicated pin. All of these features are intended to reduce required external circuitry and improve performance. Figure 8 shows the ADS1015 functional block diagram.

The ADS1013/4/5 A/D core measures a differential signal, V_{IN}, that is the difference of AIN_P and AIN_N. A MUX is available on the ADS1015. This architecture results in a very strong attenuation of any common-mode signals. The converter core consists of a differential, switched-capacitor ΔΣ modulator followed by a digital filter. Input signals are compared to the internal voltage reference. The digital filter receives a high-speed bitstream from the modulator and outputs a code proportional to the input voltage.

The ADS1013/4/5 have two available conversion modes: single-shot mode and continuous conversion mode. In single-shot mode, the ADC performs one conversion of the input signal upon request and stores the value to an internal result register. The device then enters a low-power shutdown mode. This mode is intended to provide significant power savings in systems that only require periodic conversions or when there are long idle periods between conversions. In continuous conversion mode, the ADC automatically begins a conversion of the input signal as soon as the previous conversion is completed. The rate of continuous conversion is equal to the programmed data rate. Data can be read at any time and always reflect the most recent completed conversion.

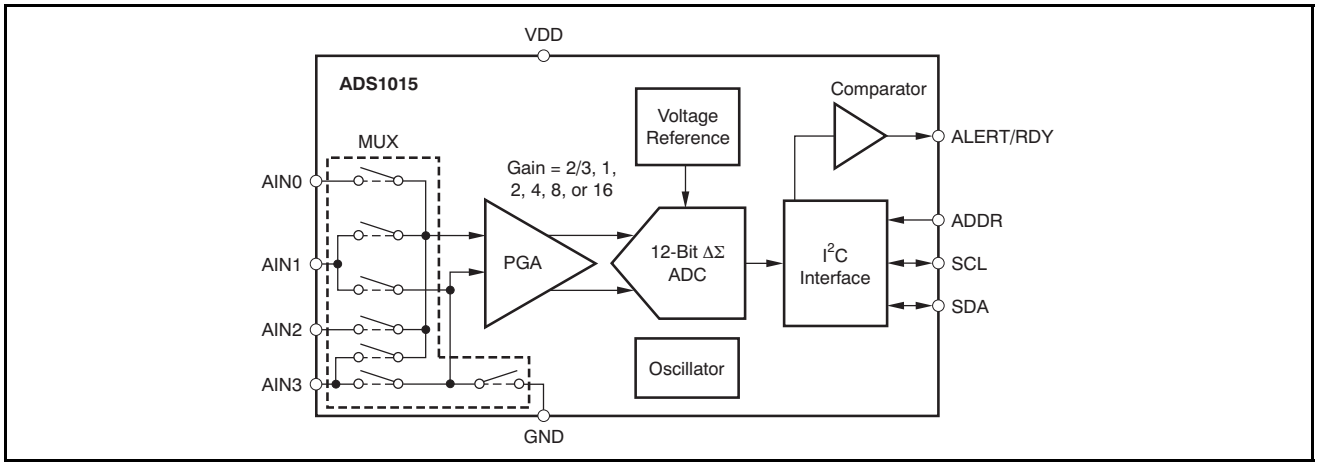


Figure 8. ADS1015 Functional Block Diagram

QUICKSTART GUIDE

This section provides a brief example of ADS1013/4/5 communications. Refer to subsequent sections of this data sheet for more detailed explanations. Hardware for this design includes: one ADS1013/4/5 configured with an I^2C address of 1001000; a microcontroller with an I^2C interface (TI recommends the MSP430 product line); discrete components such as resistors, capacitors, and serial connectors; and a 2V to 5V power supply. Figure 9 shows the basic hardware configuration.

The ADS1013/4/5 communicate with the master (microcontroller) through an I^2C interface. The master provides a clock signal on the SCL pin and data are transferred via the SDA pin. The ADS1013/4/5 never drive the SCL pin. For information on programming and debugging the microcontroller being used, refer to the device-specific product data sheet.

The first byte sent by the master should be the ADS1013/4/5 address followed by a bit that instructs the ADS1013/4/5 to listen for a subsequent byte. The second byte is the register pointer. Refer to Table 6 for a register map. The third and fourth bytes sent from the master are written to the register indicated in the second byte. Refer to Figure 16 and Figure 17 for read and write operation timing diagrams, respectively. All read and write transactions with the ADS1013/4/5 must be preceded by a start condition and followed by a stop condition.

For example, to write to the configuration register to set the ADS1013/4/5 to continuous conversion mode and then read the conversion result, send the following bytes in this order:

Write to Config register:

First byte: 0b10010000 (first 7-bit I^2C address followed by a low read/write bit)

Second byte: 0b00000001 (points to Config register)

Third byte: 0b00000100 (MSB of the Config register to be written)

Fourth byte: 0b10000011 (LSB of the Config register to be written)

Write to Pointer register:

First byte: 0b10010000 (first 7-bit I^2C address followed by a low read/write bit)

Second byte: 0b00000000 (points to Conversion register)

Read Conversion register:

First byte: 0b10010001 (first 7-bit I^2C address followed by a high read/write bit)

Second byte: the ADS1013/4/5 response with the MSB of the Conversion register

Third byte: the ADS1013/4/5 response with the LSB of the Conversion register

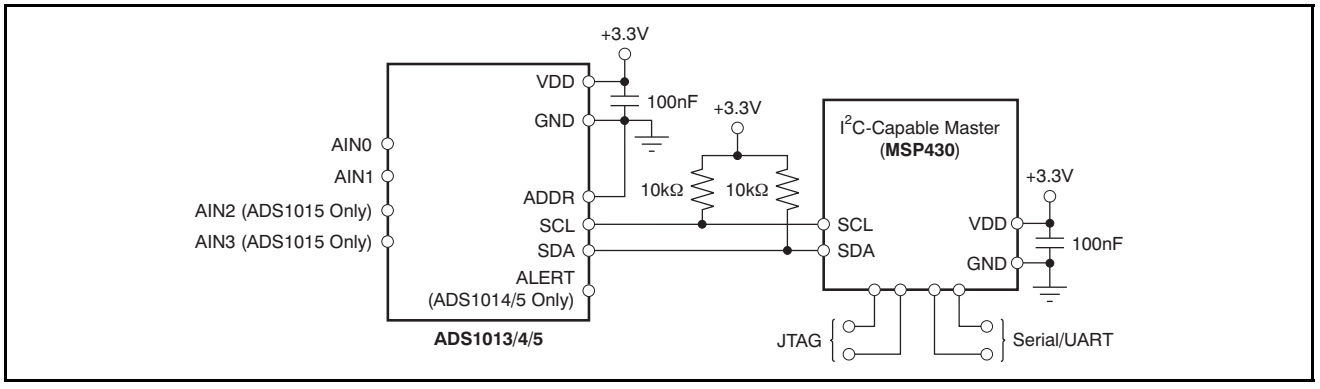


Figure 9. Basic Hardware Configuration

Copyright © 2009, Texas Instruments Incorporated

MULTIPLEXER

The ADS1015 contains an input multiplexer, as shown in Figure 10. Either four single-ended or two differential signals can be measured. Additionally, AIN0 and AIN1 may be measured differentially to AIN3. The multiplexer is configured by three bits in the Config register. When single-ended signals are measured, the negative input of the ADC is internally connected to GND by a switch within the multiplexer.

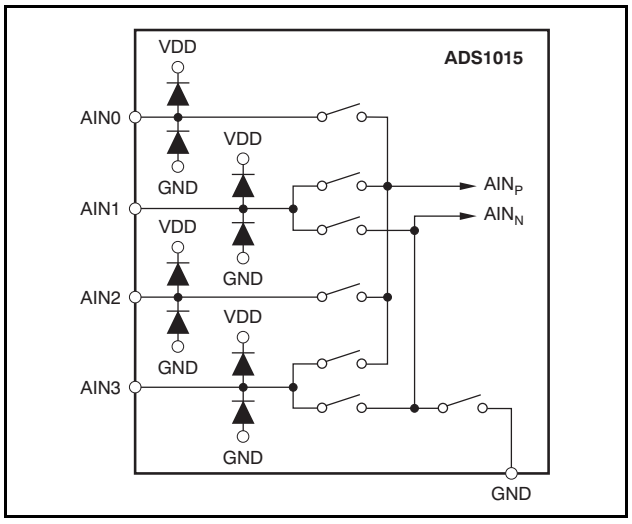


Figure 10. ADS1015 MUX

The ADS1013 and ADS1014 do not have a multiplexer. Either one differential or one single-ended signal may be measured with these devices. For single-ended measurements, connect the AIN1 pin to GND. Note that in subsequent sections of this data sheet, AIN_P refers to AIN0 and AIN_N refers to AIN1 for the ADS1013 and ADS1014.

When measuring single-ended inputs it is important to note that the negative range of the output codes are not used. These codes are for measuring negative differential signals such as $(AIN_P - AIN_N) < 0$. ESD diodes to VDD and GND protect the inputs on all three devices (ADS1013, ADS1014, and ADS1015). To prevent the ESD diodes from turning on, the absolute voltage on any input must stay within the following range:

GND – 0.3V < AINx < VDD + 0.3V

If it is possible that the voltages on the input pins may violate these conditions, external Schottky clamp diodes and/or series resistors may be required to limit the input current to safe values (see the Absolute Maximum Ratings table).

Also, overdriving one unused input on the ADS1015 may affect conversions taking place on other input pins. If overdrive on unused inputs is possible, again it is recommended to clamp the signal with external Schottky diodes.

ANALOG INPUTS

The ADS1013/4/5 use a switched-capacitor input stage where capacitors are continuously charged and then discharged to measure the voltage between AIN_P and AIN_N. The capacitors used are small, and to external circuitry the average loading appears resistive. This structure is shown in Figure 12. The resistance is set by the capacitor values and the rate at which they are switched. Figure 11 shows the on/off setting of the switches illustrated in Figure 12. During the sampling phase, S_1 switches are closed. This event charges C_{A1} to AIN_P, C_{A2} to AIN_N, and C_B to $(AIN_P - AIN_N)$. During the discharge phase, S_1 is first opened and then S_2 is closed. Both C_{A1} and C_{A2} then discharge to approximately 0.7V and C_B discharges to 0V. This charging draws a very small transient current from the source driving the ADS1013/4/5 analog inputs. The average value of this current can be used to calculate the effective impedance (R_{eff}) where $R_{eff} = V_{IN}/I_{AVERAGE}$.

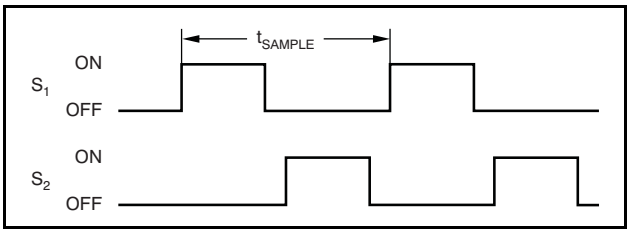


Figure 11. S_1 and S_2 Switch Timing for Figure 12

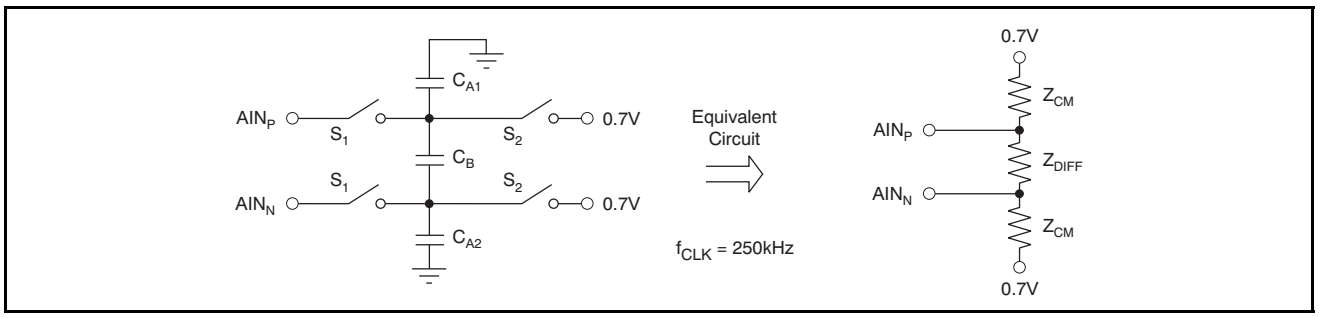


Figure 12. Simplified Analog Input Circuit

The common-mode input impedance is measured by applying a common-mode signal to shorted AIN_P and AIN_N inputs and measuring the average current consumed by each pin. The common-mode input impedance changes depending on the PGA gain setting, but is approximately 6MΩ for the default PGA gain setting. In Figure 12, the common-mode input impedance is Z_{CM}.

The differential input impedance is measured by applying a differential signal to AIN_P and AIN_N inputs where one input is held at 0.7V. The current that flows through the pin connected to 0.7V is the differential current and scales with the PGA gain setting. In Figure 12, the differential input impedance is Z_{DIFF}. Table 2 describes the typical differential input impedance.

Table 2. Differential Input Impedance

FS (V)	DIFFERENTIAL INPUT IMPEDANCE
±6.144V(1)	22MΩ
±4.096V(1)	15MΩ
±2.048V	4.9MΩ
±1.024V	2.4MΩ
±0.512V	710kΩ
±0.256V	710kΩ

1. This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.

The typical value of the input impedance cannot be neglected. Unless the input source has a low impedance, the ADS1013/4/5 input impedance may affect the measurement accuracy. For sources with high output impedance, buffering may be necessary. Active buffers introduce noise, and also introduce offset and gain errors. All of these factors should be considered in high-accuracy applications.

Because the clock oscillator frequency drifts slightly with temperature, the input impedances also drift. For many applications, this input impedance drift can be ignored, and the values given in Table 2 for typical input impedance are valid.

FULL-SCALE INPUT

A programmable gain amplifier (PGA) is implemented before the ΔΣ core of the ADS1014/5. The PGA can be set to gains of 2/3, 1, 2, 4, 8, and 16. Table 3 shows the corresponding full-scale (FS) ranges. The PGA is configured by three bits in the Config register. The ADS1013 has a fixed full-scale input range of ±2.048V. The PGA = 2/3 setting allows input measurement to extend up to the supply voltage when VDD is larger than 4V. Note though that in this case (as well as for PGA = 1 and VDD < 4V), it is not possible to reach a full-scale output code on the ADC. Analog input voltages may never exceed the analog input voltage limits given in the Electrical Characteristics table.

Table 3. PGA Gain Full-Scale Range

PGA SETTING	FS (V)
2/3	±6.144V(1)
1	±4.096V(1)
2	±2.048V
4	±1.024V
8	±0.512V
16	±0.256V

1. This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.

Copyright © 2009, Texas Instruments Incorporated

DATA FORMAT

The ADS1013/4/5 provide 12 bits of data in binary twos complement format. The positive full-scale input produces an output code of 7FF0h and the negative full-scale input produces an output code of 8000h. The output clips at these codes for signals that exceed full-scale. Table 4 summarizes the ideal output codes for different input signals. Figure 13 shows code transitions versus input voltage.

Table 4. Input Signal versus Ideal Output Code

INPUT SIGNAL, V_{IN} (AIN_P – AIN_N)	IDEAL OUTPUT CODE(1)
≥ FS (2^{11} – 1)/2^{11}	7FF0h
+FS/2^{11}	0010h
0	0
–FS/2^{11}	FFF0h
≤ –FS	8000h

1. Excludes the effects of noise, INL, offset, and gain errors.

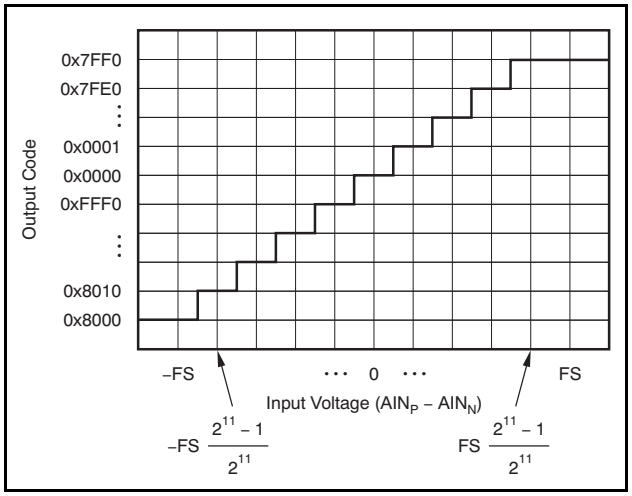


Figure 13. ADS1013/4/5 Code Transition Diagram

ALIASING

As with any data converter, if the input signal contains frequencies greater than half the data rate, aliasing occurs. To prevent aliasing, the input signal must be bandlimited. Some signals are inherently bandlimited. For example, the output of a thermocouple, which has a limited rate of change. Nevertheless, they can contain noise and interference components. These components can fold back into the sampling band in the same way as with any other signal.

The ADS1013/4/5 digital filter provides some attenuation of high-frequency noise, but the digital Sinc filter frequency response cannot completely replace an anti-aliasing filter. For a few applications, some external filtering may be needed; in such instances, a simple RC filter is adequate.

When designing an input filter circuit, be sure to take into account the interaction between the filter network and the input impedance of the ADS1013/4/5.

OPERATING MODES

The ADS1013/4/5 operate in one of two modes: continuous conversion or single-shot. In continuous conversion mode, the ADS1013/4/5 continuously perform conversions. Once a conversion has been completed, the ADS1013/4/5 place the result in the Conversion register and immediately begins another conversion. In single-shot mode, the ADS1013/4/5 wait until the OS bit is set high. Once asserted, the bit is set to '0', indicating that a conversion is currently in progress. Once conversion data are ready, the OS bit reasserts and the device powers down. Writing a '1' to the OS bit during a conversion has no effect.

RESET AND POWER-UP

When the ADS1013/4/5 powers up, a reset is performed. As part of the reset process, the ADS1013/4/5 set all of the bits in the Config register to the respective default settings.

The ADS1013/4/5 respond to the I^2C general call reset command. When the ADS1013/4/5 receive a general call reset, an internal reset is performed as if the device had been powered on.

DUTY CYCLING FOR LOW POWER

For many applications, the improved performance at low data rates may not be required. For these applications, the ADS1013/4/5 support duty cycling that can yield significant power savings by periodically requesting high data rate readings at an effectively lower data rate. For example, an ADS1013/4/5 in power-down mode with a data rate set to 3300SPS could be operated by a microcontroller that instructs a single-shot conversion every 7.8ms (128SPS). Because a conversion at 3300SPS only requires about 0.3ms, the ADS1013/4/5 enter power-down mode for the remaining 7.5ms. In this configuration, the ADS1013/4/5 consume about 1/25th the power of the ADS1013/4/5 operated in continuous conversion mode. The rate of duty cycling is completely arbitrary and is defined by the master controller. The ADS1013/4/5 offer lower data rates that do not implement duty cycling and offer improved noise performance if it is needed.

Copyright © 2009, Texas Instruments Incorporated

COMPARATOR (ADS1014/15 ONLY)

The ADS1014/5 are each equipped with a customizable comparator that can issue an alert on the ALERT/RDY pin. This feature can significantly reduce external circuitry for many applications. The comparator can be implemented as either a traditional comparator or a window comparator via the COMP_MODE bit in the Config register. When implemented as a traditional comparator, the ALERT/RDY pin asserts (active low by default) when conversion data exceed the limit set in the high threshold register. The comparator then deasserts when the input signal falls below the low threshold register value. In window comparator mode, the ALERT/RDY pin asserts if conversion data exceed the high threshold register or fall below the low threshold register.

In either window or traditional comparator mode, the comparator can be configured to latch once asserted by the COMP_LAT bit in the Config register. This setting causes the assertion to remain even if the input signal is not beyond the bounds of the threshold registers. This latched assertion can be cleared by issuing an SMBus alert response or by reading the Conversion register. The COMP_POL bit in the Config register configures the ALERT/RDY pin as active high or active low. Operational diagrams for the comparator modes are shown in Figure 14 and Figure 15.

The comparator can be configured to activate the ALERT/RDY pin after a set number of successive readings exceed the threshold. The comparator can be configured to wait for one, two, or four readings beyond the threshold before activating the ALERT/RDY pin by changing the COMP_QUE bits in the Config register. The COMP_QUE bits can also disable the comparator function.

CONVERSION READY PIN (ADS1014/5 ONLY)

The ALERT/RDY pin can also be configured as a conversion ready pin. This mode of operation can be realized if the MSB of the high threshold register is set to '1' and the MSB of the low threshold register is set to '0'. The COMP_POL bit continues to function and the COMP_QUE bits can disable the pin; however, the COMP_MODE and COMP_LAT bits no longer control any function. When configured as a conversion ready pin, ALERT/RDY continues to require a pull-up resistor. When in continuous conversion mode, the ADS1013/4/5 provide a brief (~8µs) pulse on the ALERT/RDY pin at the end of each conversion. When in single-shot shutdown mode, the ALERT/RDY pin asserts low at the end of a conversion if the COMP_POL bit is set to '0'.

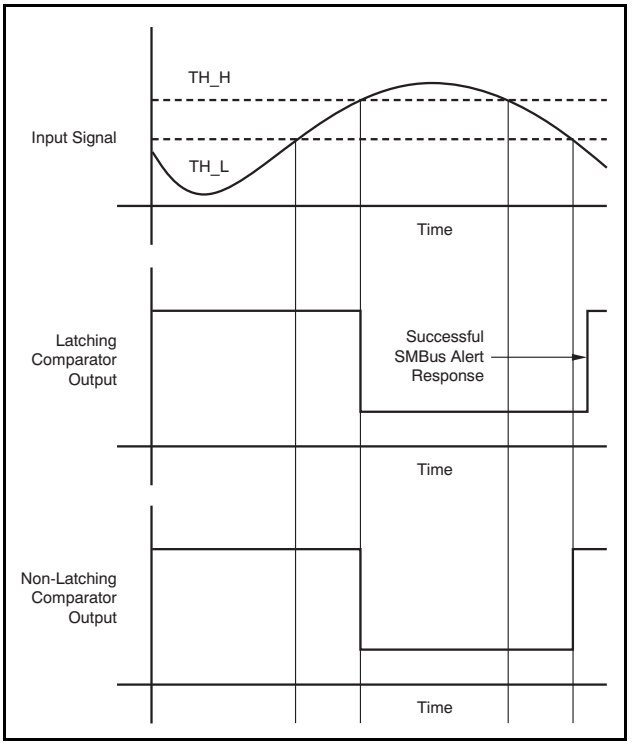


Figure 14. Alert Pin Timing Diagram When Configured as a Traditional Comparator

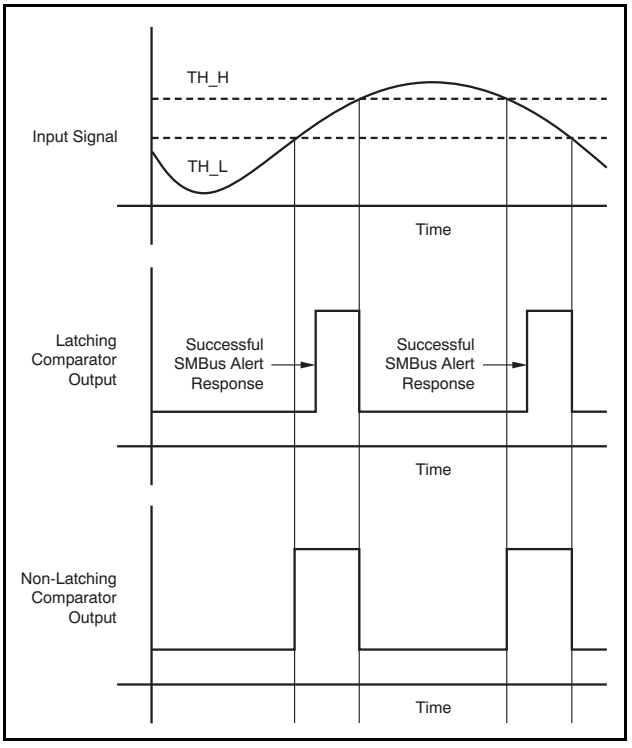


Figure 15. Alert Pin Timing Diagram When Configured as a Window Comparator

Copyright © 2009, Texas Instruments Incorporated

SMBus ALERT RESPONSE

When configured in latching mode (COMP_LAT = '1' in the Config register), the ALERT/RDY pin can be implemented with an SMBus alert. The pin asserts if the comparator detects a conversion that exceeds an upper or lower threshold. This interrupt is latched and can be cleared only by reading conversion data, or by issuing a successful SMBus alert response and reading the asserting device I^2C address. If conversion data exceed the upper or lower thresholds after being cleared, the pin reasserts. This assertion does not affect conversions that are already in progress. The ALERT/RDY pin, as with the SDA pin, is an open-drain pin. This architecture allows several devices to share the same interface bus. When disabled, the pin holds a high state so that it does not interfere with other devices on the same bus line.

When the master senses that the ALERT/RDY pin has latched, it issues an SMBus alert command (00011001) to the I^2C bus. Any ADS1014/5 data converters on the I^2C bus with the ALERT/RDY pins asserted respond to the command with the slave address. This sequence is illustrated in Figure 18. In the event that two or more ADS1014/5 data converters present on the bus assert the latched ALERT/RDY pin, arbitration during the address response portion of the SMBus alert decides which device clears its assertion. The device with the lowest I^2C address always wins arbitration. If a device loses arbitration, it does not clear the comparator output pin assertion. The master then repeats the SMBus alert response until all devices have had the respective assertions cleared. In window comparator mode, the SMBus alert status bit indicates a '1' if signals exceed the high threshold and a '0' if signals exceed the low threshold.

I^2C INTERFACE

The ADS1013/4/5 communicate through an I^2C interface. I^2C is a two-wire open-drain interface that supports multiple devices and masters on a single bus. Devices on the I^2C bus only drive the bus lines low by connecting them to ground; they never drive the bus lines high. Instead, the bus wires are pulled high by pull-up resistors, so the bus wires are high when no device is driving them low. This way, two devices cannot conflict; if two devices drive the bus simultaneously, there is no driver contention.

Communication on the I^2C bus always takes place between two devices, one acting as the master and the other as the slave. Both masters and slaves can read and write, but slaves can only do so under the direction of the master. Some I^2C devices can act as masters or slaves, but the ADS1013/4/5 can only act as slave devices.

An I^2C bus consists of two lines, SDA and SCL. SDA carries data; SCL provides the clock. All data are transmitted across the I^2C bus in groups of eight bits. To send a bit on the I^2C bus, the SDA line is driven to the appropriate level while SCL is low (a low on SDA indicates the bit is zero; a high indicates the bit is one). Once the SDA line settles, the SCL line is brought high, then low. This pulse on SCL clocks the SDA bit into the receiver shift register. If the I^2C bus is held idle for more than 25ms, the bus times out.

The I^2C bus is bidirectional: the SDA line is used for both transmitting and receiving data. When the master reads from a slave, the slave drives the data line; when the master sends to a slave, the master drives the data line. The master always drives the clock line. The ADS1013/4/5 never drive SCL, because they cannot act as a master. On the ADS1013/4/5, SCL is an input only.

Most of the time the bus is idle; no communication occurs, and both lines are high. When communication is taking place, the bus is active. Only master devices can start a communication and initiate a START condition on the bus. Normally, the data line is only allowed to change state while the clock line is low. If the data line changes state while the clock line is high, it is either a START condition or a STOP condition. A START condition occurs when the clock line is high and the data line goes from high to low. A STOP condition occurs when the clock line is high and the data line goes from low to high.

After the master issues a START condition, it sends a byte that indicates which slave device it wants to communicate with. This byte is called the *address byte*. Each device on an I^2C bus has a unique 7-bit address to which it responds. The master sends an address in the address byte, together with a bit that indicates whether it wishes to read from or write to the slave device.

Every byte transmitted on the I^2C bus, whether it is address or data, is acknowledged with an *acknowledge* bit. When the master has finished sending a byte (eight data bits) to a slave, it stops driving SDA and waits for the slave to acknowledge the byte. The slave acknowledges the byte by pulling SDA low. The master then sends a clock pulse to clock the acknowledge bit. Similarly, when the master has finished reading a byte, it pulls SDA low to acknowledge this to the slave. It then sends a clock pulse to clock the bit. (The master always drives the clock line.)

A *not-acknowledge* is performed by simply leaving SDA high during an acknowledge cycle. If a device is not present on the bus, and the master attempts to address it, it receives a not-acknowledge because no device is present at that address to pull the line low.

Copyright © 2009, Texas Instruments Incorporated

When the master has finished communicating with a slave, it may issue a STOP condition. When a STOP condition is issued, the bus becomes idle again. The master may also issue another START condition. When a START condition is issued while the bus is active, it is called a repeated START condition.

See the *Timing Requirements* section for a timing diagram showing the ADS1013/4/5 I²C transaction.

I²C ADDRESS SELECTION

The ADS1013/4/5 have one address pin, ADDR, that sets the I²C address. This pin can be connected to ground, VDD, SDA, or SCL, allowing four addresses to be selected with one pin as shown in Table 5. The state of the address pin ADDR is sampled continuously.

Table 5. ADDR Pin Connection and Corresponding Slave Address

ADDR PIN	SLAVE ADDRESS
Ground	1001000
VDD	1001001
SDA	1001010
SCL	1001011

I²C GENERAL CALL

The ADS1013/4/5 respond to the I²C general call address (0000000) if the eighth bit is '0'. The devices acknowledge the general call address and respond to commands in the second byte. If the second byte is 00000110 (06h), the ADS1013/4/5 reset the internal registers and enter power-down mode.

I²C SPEED MODES

The I²C bus operates at one of three speeds. Standard mode allows a clock frequency of up to 100kHz; fast mode permits a clock frequency of up to 400kHz; and high-speed mode (also called Hs mode) allows a clock frequency of up to 3.4MHz. The ADS1013/4/5 are fully compatible with all three modes.

No special action is required to use the ADS1013/4/5 in standard or fast mode, but high-speed mode must be activated. To activate high-speed mode, send a special address byte of *00001xxx* following the START condition, where *xxx* are bits unique to the Hs-capable master. This byte is called the Hs master code. (Note that this is different from normal address bytes; the eighth bit does not indicate read/write status.) The ADS1013/4/5 do not acknowledge this byte; the I²C specification prohibits acknowledgment of the Hs master code. Upon receiving a master code, the ADS1013/4/5 switch on Hs mode filters, and communicate at up to 3.4MHz. The ADS1013/4/5 switch out of Hs mode with the next STOP condition.

For more information on high-speed mode, consult the I²C specification.

SLAVE MODE OPERATIONS

The ADS1013/4/5 can act as either slave receivers or slave transmitters. As a slave device, the ADS1013/4/5 cannot drive the SCL line.

Receive Mode:

In slave receive mode the first byte transmitted from the master to the slave is the address with the R/$\overline{W}$ bit low. This byte allows the slave to be written to. The next byte transmitted by the master is the register pointer byte. The ADS1013/4/5 then acknowledge receipt of the register pointer byte. The next two bytes are written to the address given by the register pointer. The ADS1013/4/5 acknowledge each byte sent. Register bytes are sent with the most significant byte first, followed by the least significant byte.

Transmit Mode:

In slave transmit mode, the first byte transmitted by the master is the 7-bit slave address followed by the high R/$\overline{W}$ bit. This byte places the slave into transmit mode and indicates that the ADS1013/4/5 are being read from. The next byte transmitted by the slave is the most significant byte of the register that is indicated by the register pointer. This byte is followed by an acknowledgment from the master. The remaining least significant byte is then sent by the slave and is followed by an acknowledgment from the master. The master may terminate transmission after any byte by not acknowledging or issuing a START or STOP condition.

WRITING/READING THE REGISTERS

To access a specific register from the ADS1013/4/5, the master must first write an appropriate value to the Pointer register. The Pointer register is written directly after the slave address byte, low R/$\overline{W}$ bit, and a successful slave acknowledgment. After the Pointer register is written, the slave acknowledges and the master issues a STOP or a repeated START condition.

Copyright © 2009, Texas Instruments Incorporated

When reading from the ADS1013/4/5, the previous value written to the Pointer register determines the register that is read from. To change which register is read, a new value must be written to the Pointer register. To write a new value to the Pointer register, the master issues a slave address byte with the R/$\overline{W}$ bit low, followed by the Pointer register byte. No additional data need to be transmitted, and a STOP condition can be issued by the master. The master may now issue a START condition and send the slave address byte with the R/$\overline{W}$ bit high to begin the read. Figure 16 details this sequence. If repeated reads from the same register are desired, there is no need to continually send Pointer register bytes, because the ADS1013/4/5 store the value of the Pointer register until it is modified by a write operation. However, every write operation requires the Pointer register to be written.

REGISTERS

The ADS1013/4/5 have four registers that are accessible via the I^2C port. The Conversion register contains the result of the last conversion. The Config register allows the user to change the ADS1013/4/5 operating modes and query the status of the devices. Two registers, Lo_thresh and Hi_thresh, set the threshold values used for the comparator function.

POINTER REGISTER

The four registers are accessed by writing to the Pointer register byte; see Figure 16. Table 6 and Table 7 indicate the Pointer register byte map.

Table 6. Register Address

BIT 1	BIT 0	REGISTER
0	0	Conversion register
0	1	Config register
1	0	Lo_thresh register
1	1	Hi_thresh register

CONVERSION REGISTER

The 16-bit register contains the result of the last conversion in binary twos complement format. Following reset or power-up, the Conversion register is cleared to '0', and remains '0' until the first conversion is completed.

The register format is shown in Table 8.

CONFIG REGISTER

The 16-bit register can be used to control the ADS1013/4/5 operating mode, input selection, data rate, PGA settings, and comparator modes. The register format is shown in Table 9.

Table 7. Pointer Register Byte (Write-Only)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	0	0	0	0	0	Register address	

Table 8. Conversion Register (Read-Only)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

Table 9. Config Register (Read/Write)

BIT	15	14	13	12	11	10	9	8
NAME	OS	MUX2	MUX1	MUX0	PGA2	PGA1	PGA0	MODE
BIT	7	6	5	4	3	2	1	0
NAME	DR2	DR1	DR0	COMP_MODE	COMP_POL	COMP_LAT	COMP_QUE1	COMP_QUE0

Default = 8583h.

Bit [15] **OS: Operational status/single-shot conversion start**

This bit determines the operational status of the device.
This bit can only be written when in power-down mode.

For a write status:
0 : No effect
1 : Begin a single conversion (when in power-down mode)
For a read status:
0 : Device is currently performing a conversion
1 : Device is not currently performing a conversion

Copyright © 2009, Texas Instruments Incorporated

Bits [14:12] **MUX[2:0]: Input multiplexer configuration (ADS1015 only)**

These bits configure the input multiplexer. They serve no function on the ADS1013/4.

000 : AIN_P = AIN0 and AIN_N = AIN1 (default)	100 : AIN_P = AIN0 and AIN_N = GND
001 : AIN_P = AIN0 and AIN_N = AIN3	101 : AIN_P = AIN1 and AIN_N = GND
010 : AIN_P = AIN1 and AIN_N = AIN3	110 : AIN_P = AIN2 and AIN_N = GND
011 : AIN_P = AIN2 and AIN_N = AIN3	111 : AIN_P = AIN3 and AIN_N = GND

Bits [11:9] **PGA[2:0]: Programmable gain amplifier configuration (ADS1014 and ADS1015 only)**

These bits configure the programmable gain amplifier. They serve no function on the ADS1013.

000 : FS = ±6.144V(1)	100 : FS = ±0.512V
001 : FS = ±4.096V(1)	101 : FS = ±0.256V
010 : FS = ±2.048V (default)	110 : FS = ±0.256V
011 : FS = ±1.024V	111 : FS = ±0.256V

Bit [8] **MODE: Device operating mode**

This bit controls the current operational mode of the ADS1013/4/5.
0 : Continuous conversion mode
1 : Power-down single-shot mode (default)

Bits [7:5] **DR[2:0]: Data rate**

These bits control the data rate setting.

000 : 128SPS	100 : 1600SPS (default)
001 : 250SPS	101 : 2400SPS
010 : 490SPS	110 : 3300SPS
011 : 920SPS	111 : 3300SPS

Bit [4] **COMP_MODE: Comparator mode (ADS1014 and ADS1015 only)**

This bit controls the comparator mode of operation. It changes whether the comparator is implemented as a traditional comparator (COMP_MODE = '0') or as a window comparator (COMP_MODE = '1'). It serves no function on the ADS1013.
0 : Traditional comparator with hysteresis (default)
1 : Window comparator

Bit [3] **COMP_POL: Comparator polarity (ADS1014 and ADS1015 only)**

This bit controls the polarity of the ALERT/RDY pin. When COMP_POL = '0' the comparator output is active low. When COMP_POL='1' the ALERT/RDY pin is active high. It serves no function on the ADS1013.
0 : Active low (default)
1 : Active high

Bit [2] **COMP_LAT: Latching comparator (ADS1014 and ADS1015 only)**

This bit controls whether the ALERT/RDY pin latches once asserted or clears once conversions are within the margin of the upper and lower threshold values. When COMP_LAT = '0', the ALERT/RDY pin does not latch when asserted. When COMP_LAT = '1', the asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master, the device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line. This bit serves no function on the ADS1013.
0 : Non-latching comparator (default)
1 : Latching comparator

Bits [1:0] **COMP_QUE: Comparator queue and disable (ADS1014 and ADS1015 only)**

These bits perform two functions. When set to '11', they disable the comparator function and put the ALERT/RDY pin into a high state. When set to any other value, they control the number of successive conversions exceeding the upper or lower thresholds required before asserting the ALERT/RDY pin. They serve no function on the ADS1013.
00 : Assert after one conversion
01 : Assert after two conversions
10 : Assert after four conversions
11 : Disable comparator (default)

(1) This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.

Copyright © 2009, Texas Instruments Incorporated

Lo_thresh AND Hi_thresh REGISTERS

The upper and lower threshold values used by the comparator are stored in two 16-bit registers. These registers store values in the same format that the output register displays values; that is, they are stored in twos complement format. Because it is implemented as a digital comparator, special attention should be taken to readjust values whenever PGA settings are changed.

A secondary conversion ready function of the comparator output pin can be realized by setting the Hi_thresh register MSB to '1' and the Lo_thresh register MSB to '0'. However, in all other cases, the Hi_thresh register must be larger than the Lo_thresh register. The threshold register formats are shown in Table 10. When set to RDY mode, the ALERT/RDY pin outputs the state of the OS bit when in single-shot mode and pulses when in continuous conversion mode. Bits [3:0] in both the Lo_thresh and Hi_thresh registers have no effect on the comparator level thresholds. These bits should be considered as *don't care* bits.

Table 10. Lo_thresh and Hi_thresh Registers

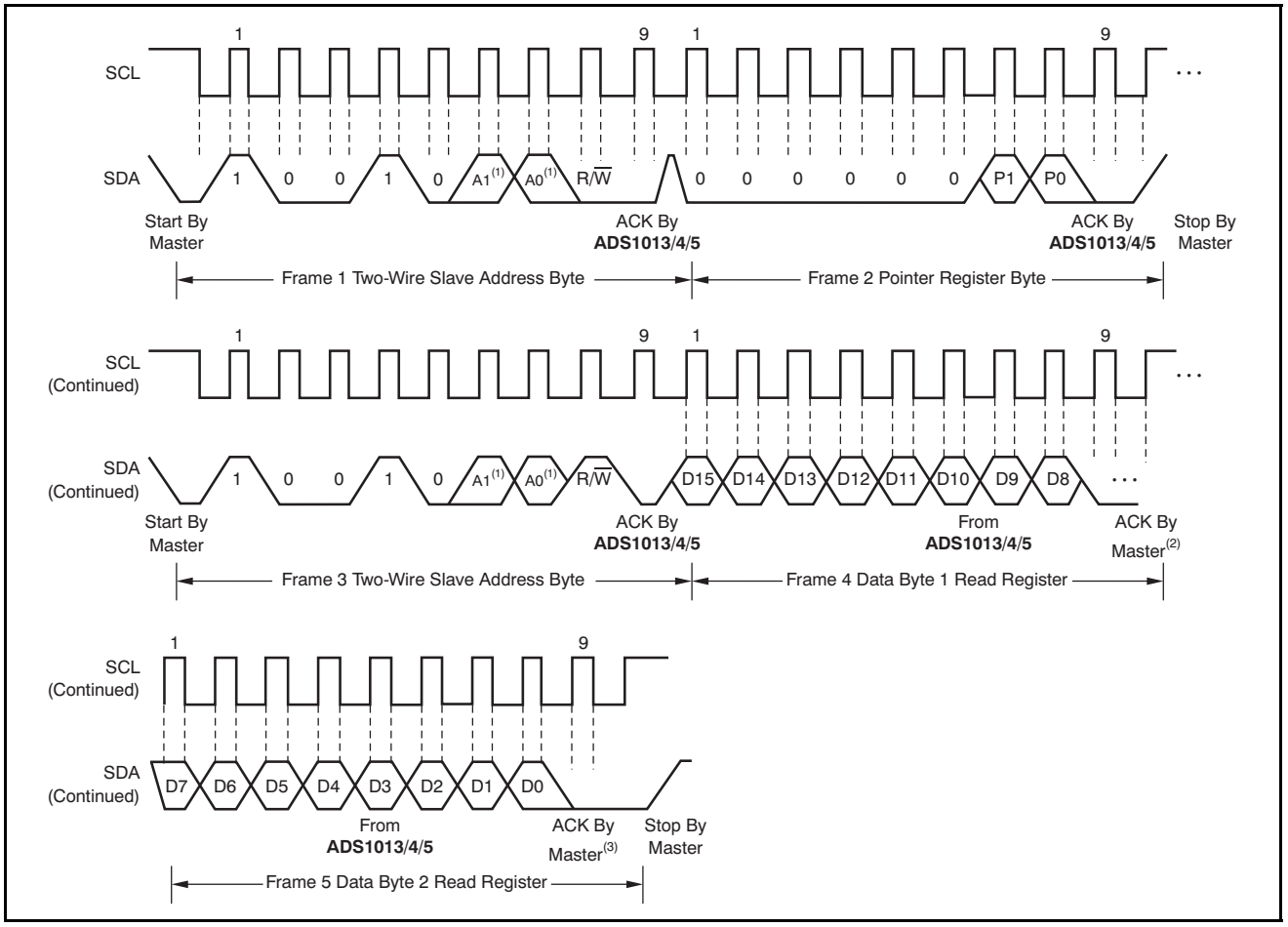
REGISTER	Lo_thresh (Read/Write)							
BIT	15	14	13	12	11	10	9	8
NAME	Lo_thresh11	Lo_thresh10	Lo_thresh9	Lo_thresh8	Lo_thresh7	Lo_thresh6	Lo_thresh5	Lo_thresh4
BIT	7	6	5	4	3	2	1	0
NAME	Lo_thresh3	Lo_thresh2	Lo_thresh1	Lo_thresh0	0	0	0	0

REGISTER	Hi_thresh (Read/Write)							
BIT	15	14	13	12	11	10	9	8
NAME	Hi_thresh11	Hi_thresh10	Hi_thresh9	Hi_thresh8	Hi_thresh7	Hi_thresh6	Hi_thresh5	Hi_thresh4
BIT	7	6	5	4	3	2	1	0
NAME	Hi_thresh3	Hi_thresh2	Hi_thresh1	Hi_thresh0	1	1	1	1

Lo_thresh default = 8000h.

Hi_thresh default = 7FFFh.

Copyright © 2009, Texas Instruments Incorporated



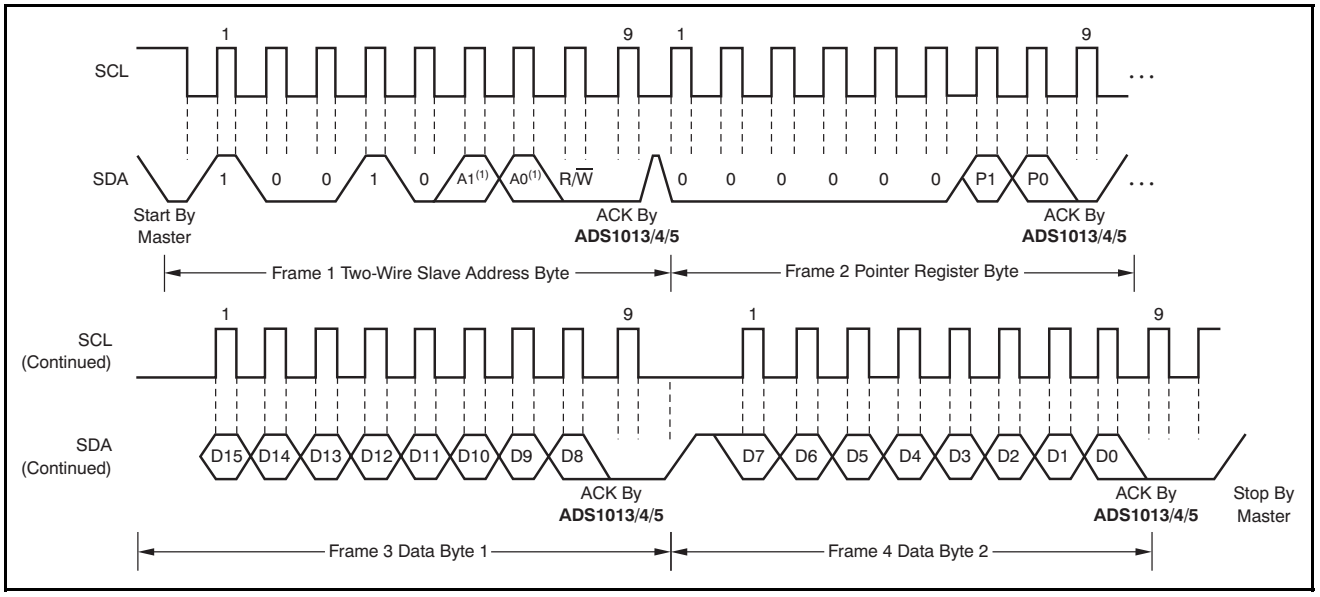
(1) The values of A0 and A1 are determined by the ADDR pin.

(2) Master can leave SDA high to terminate a single-byte read operation.

(3) Master can leave SDA high to terminate a two-byte read operation.

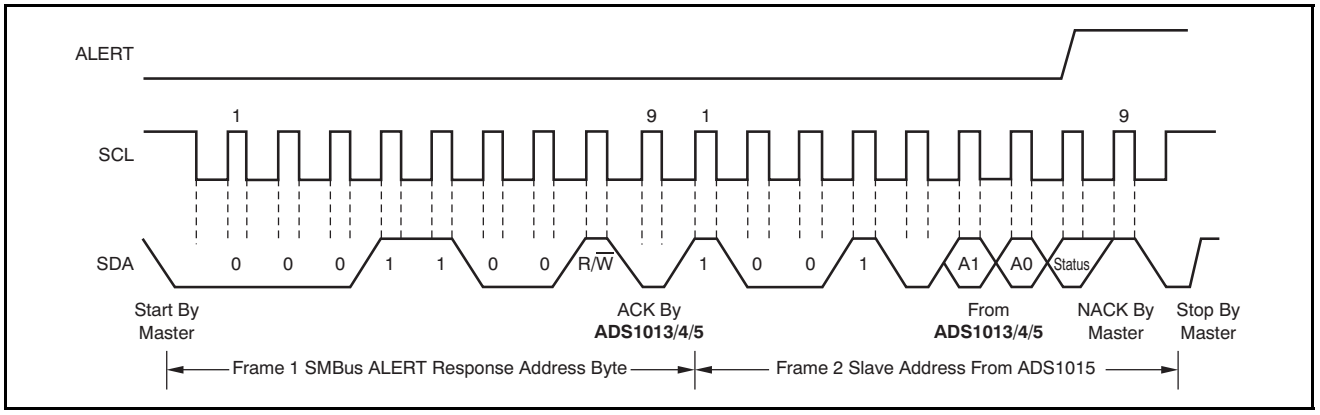
Figure 16. Two-Wire Timing Diagram for Read Word Format

Copyright © 2009, Texas Instruments Incorporated



(1) The values of A0 and A1 are determined by the ADDR pin.

Figure 17. Two-Wire Timing Diagram for Write Word Format



(1) The values of A0 and A1 are determined by the ADDR pin.

Figure 18. Timing Diagram for SMBus ALERT Response

APPLICATION INFORMATION

The following sections give example circuits and suggestions for using the ADS1013/4/5 in various situations.

BASIC CONNECTIONS

For many applications, connecting the ADS1013/4/5 is simple. A basic connection diagram for the ADS1015 is shown in Figure 19.

The fully differential voltage input of the ADS1013/4/5 is ideal for connection to differential sources with moderately low source impedance, such as thermocouples and thermistors. Although the ADS1013/4/5 can read bipolar differential signals, they cannot accept negative voltages on either input. It may be helpful to think of the ADS1013/4/5 positive voltage input as *noninverting*, and of the negative input as *inverting.*

When the ADS1013/4/5 are converting data, they draw current in short spikes. The 0.1µF bypass capacitor supplies the momentary bursts of extra current needed from the supply.

The ADS1013/4/5 interface directly to standard mode, fast mode, and high-speed mode I^2C controllers. Any microcontroller I^2C peripheral, including master-only and non-multiple-master I^2C peripherals, can operate with the ADS1013/4/5. The ADS1013/4/5 do not perform clock-stretching (that is, they never pull the clock line low), so it is not necessary to provide for this function unless other clock-stretching devices are on the same I^2C bus.

Pull-up resistors are required on both the SDA and SCL lines because I^2C bus drivers are open-drain. The size of these resistors depends on the bus operating speed and capacitance of the bus lines. Higher-value resistors consume less power, but increase the transition times on the bus, limiting the bus speed. Lower-value resistors allow higher speed at the expense of higher power consumption. Long bus lines have higher capacitance and require smaller pull-up resistors to compensate. The resistors should not be too small; if they are, the bus drivers may not be able to pull the bus lines low.

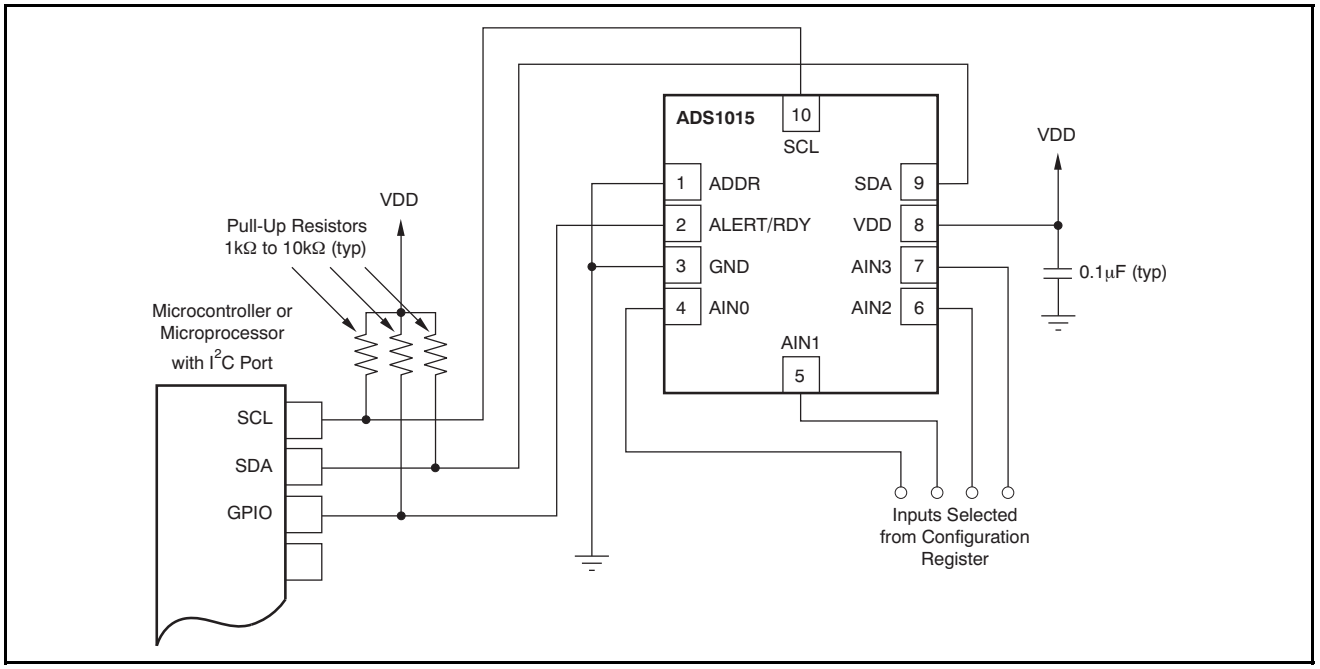


Figure 19. Typical Connections of the ADS1015

Copyright © 2009, Texas Instruments Incorporated

CONNECTING MULTIPLE DEVICES

Connecting multiple ADS1013/4/5s to a single bus is simple. Using the address pin, the ADS1013/4/5 can be set to one of four different I^2C addresses. An example showing four ADS1013/4/5 devices is given in Figure 21. Up to four ADS1013/4/5s (using different address pin configurations) can be connected to a single bus.

Note that only one set of pull-up resistors is needed per bus. The pull-up resistor values may need to be lowered slightly to compensate for the additional bus capacitance presented by multiple devices and increased line length.

The TMP421 and DAC8574 devices detect the respective I^2C bus addresses based on the states of pins. In Figure 22, the TMP421 has the address 0101010, and the DAC8574 has the address 1001100. Consult the DAC8574 and TMP421 data sheets, available at www.ti.com, for further details.

USING GPIO PORTS FOR COMMUNICATION

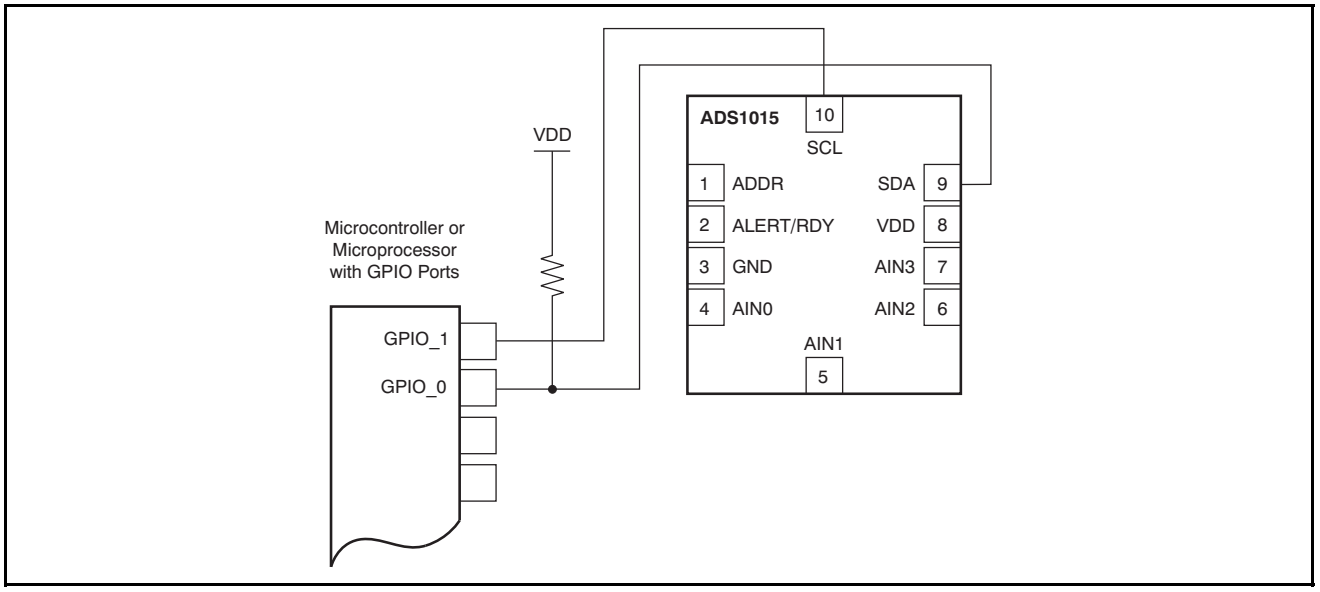
Most microcontrollers have programmable input/output (I/O) pins that can be set in software to act as inputs or outputs. If an I^2C controller is not available, the ADS1013/4/5 can be connected to GPIO pins and the I^2C bus protocol simulated, or *bit-banged*, in software. An example of this configuration for a single ADS1013/4/5 is shown in Figure 20.

Bit-banging I^2C with GPIO pins can be done by setting the GPIO line to '0' and toggling it between input and output modes to apply the proper bus states. To drive the line low, the pin is set to output '0'; to let the line go high, the pin is set to input. When the pin is set to input, the state of the pin can be read; if another device is pulling the line low, this configuration reads as a '0' in the port input register.

Note that no pull-up resistor is shown on the SCL line. In this simple case, the resistor is not needed; the microcontroller can simply leave the line on output, and set it to '1' or '0' as appropriate. This action is possible because the ADS1013/4/5 never drive the clock line low. This technique can also be used with multiple devices, and has the advantage of lower current consumption as a result of the absence of a resistive pull-up.

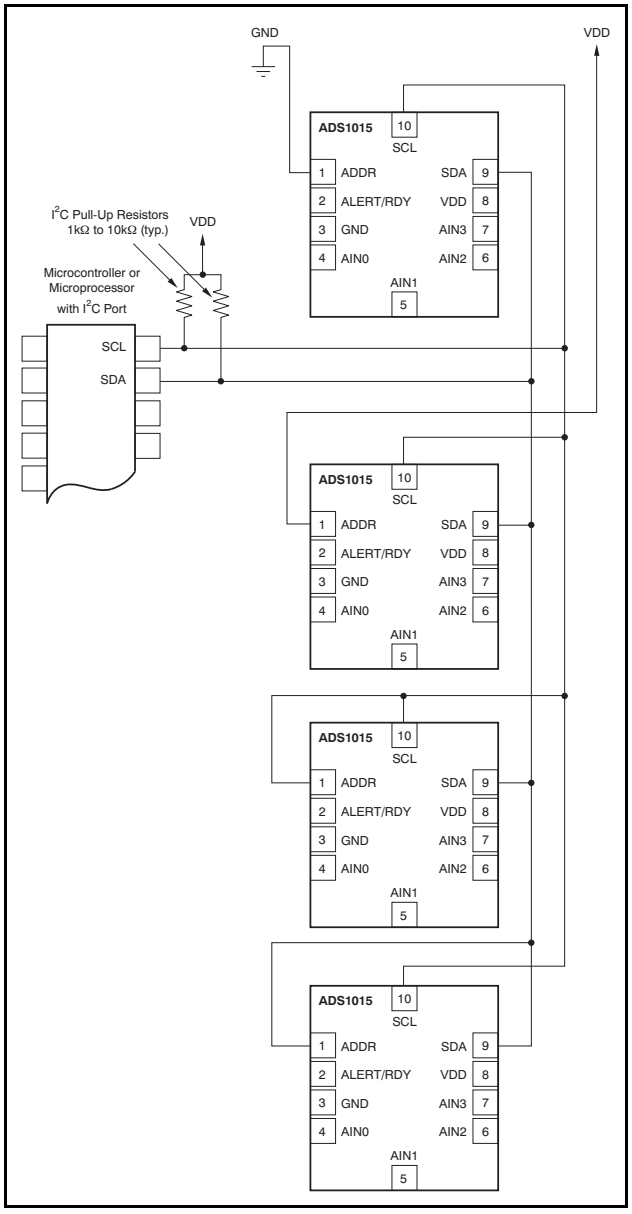
If there are any devices on the bus that may drive the clock lines low, this method should not be used; the SCL line should be high-Z or '0' and a pull-up resistor provided as usual.

Some microcontrollers have selectable strong pull-up circuits built in to the GPIO ports. In some cases, these circuits can be switched on and used in place of an external pull-up resistor. Weak pull-ups are also provided on some microcontrollers, but usually these are too weak for I^2C communication. If there is any doubt about the matter, test the circuit before committing it to production.



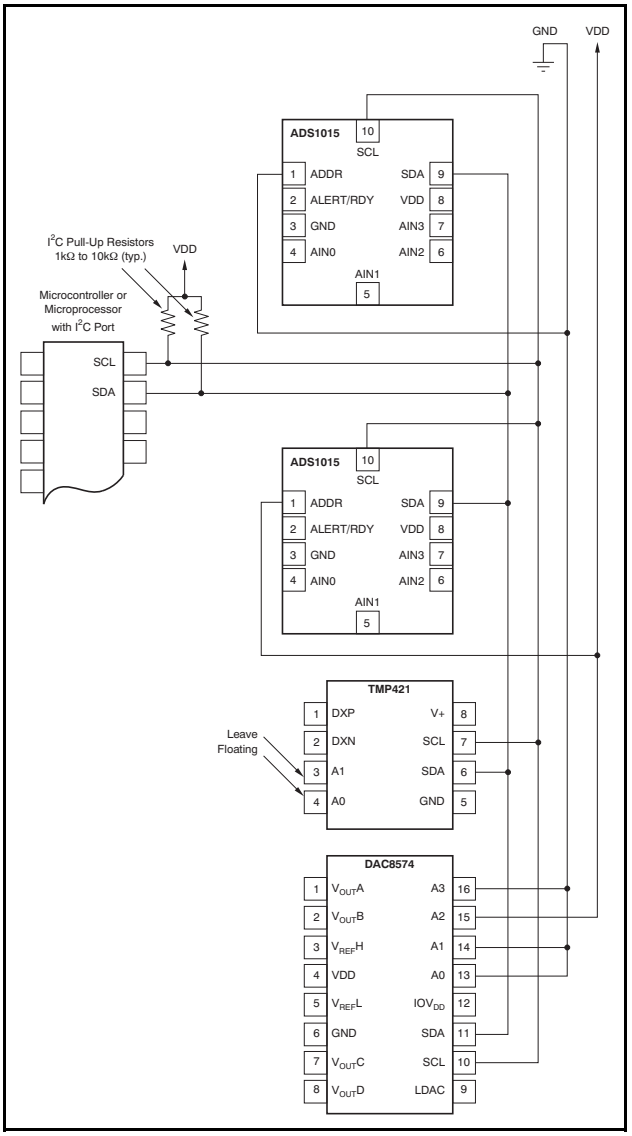
NOTE: ADS1013/4/5 power and input connections omitted for clarity.

Figure 20. Using GPIO with a Single ADS1015



NOTE: ADS1013/4/5 power and input connections omitted for clarity. The ADDR pin selects the I^2C address.

Figure 21. Connecting Multiple ADS1013/4/5s



NOTE: ADS1013/4/5 power and input connections omitted for clarity. ADDR, A3, A2, A1, and A0 select the I^2C addresses.

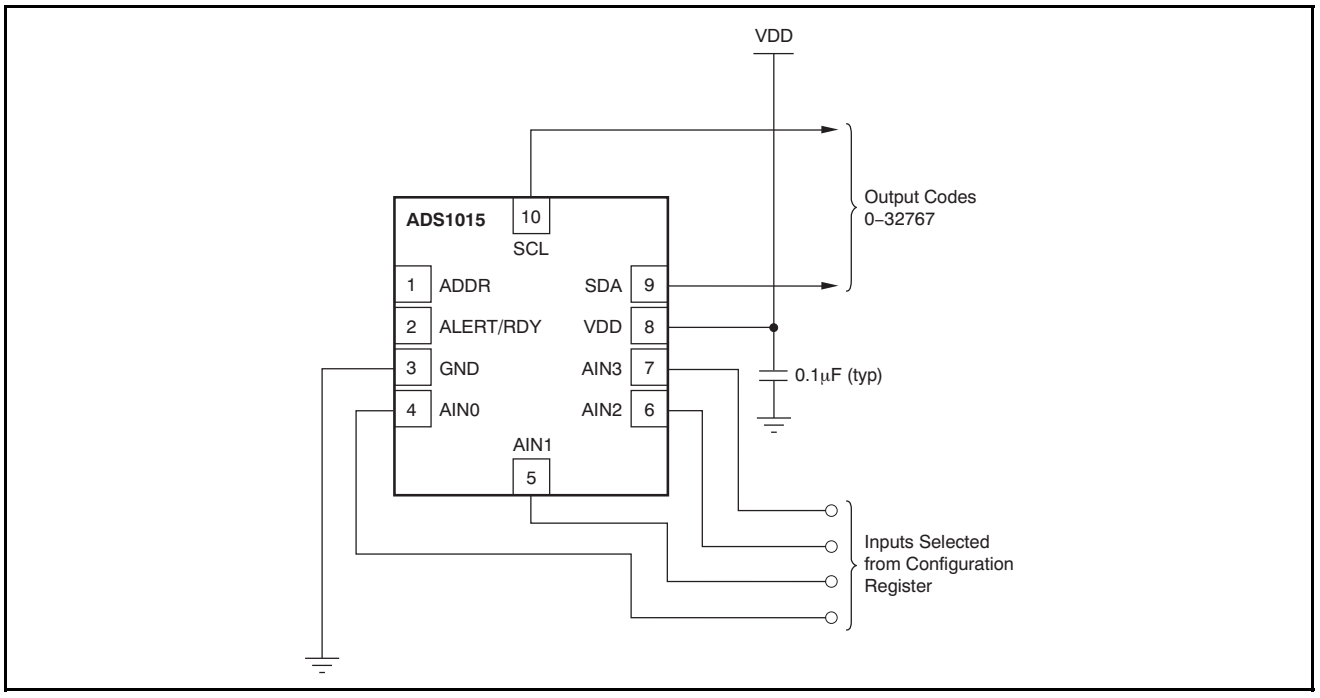
Figure 22. Connecting Multiple Device Types

Copyright © 2009, Texas Instruments Incorporated

SINGLE-ENDED INPUTS

Although the ADS1015 has two differential inputs, the device can easily measure four single-ended signals. Figure 23 shows a single-ended connection scheme. The ADS1015 is configured for single-ended measurement by configuring the MUX to measure each channel with respect to ground. Data are then read out of one input based on the selection on the configuration register. The single-ended signal can range from 0V to supply. The ADS1015 loses no linearity anywhere within the input range. Negative voltages cannot be applied to this circuit because the ADS1015 can only accept positive voltages.

The ADS1015 input range is bipolar differential with respect to the reference. The single-ended circuit shown in Figure 23 covers only half the ADS1015 input scale because it does not produce differentially negative inputs; therefore, one bit of resolution is lost.



NOTE: Digital and address pin connections omitted for clarity.

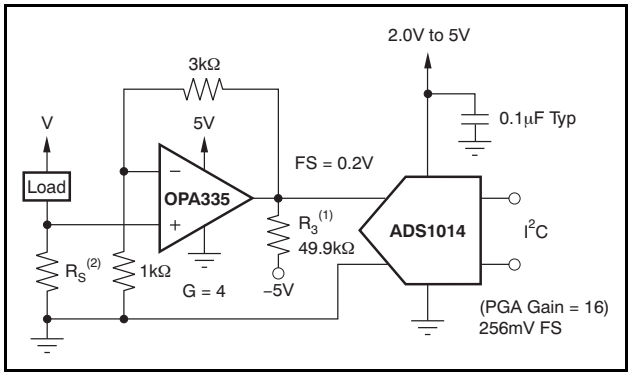
Figure 23. Measuring Single-Ended Inputs

Copyright © 2009, Texas Instruments Incorporated

LOW-SIDE CURRENT MONITOR

Figure 24 shows a circuit for a low-side shunt-type current monitor. The circuit monitors the voltage across a shunt resistor, which is sized as small as possible while giving a measurable output voltage. This voltage is amplified by an OPA335 low-drift op amp, and the result is read by the ADS1014/5.

It is suggested that the ADS1014/5 be operated at a gain of 16. The gain of the OPA335 can then be set lower. For a gain of 16, the op amp should be set up to give a maximum output voltage no greater than 0.256V. If the shunt resistor is sized to provide a maximum voltage drop of 50mV at full-scale current, the full-scale input to the ADS1014/5 is 0.2V.



(1) Pull-down resistor to allow accurate swing to 0V.

(2) R_S is sized for a 50mV drop at full-scale current.

Figure 24. Low-Side Current Measurement

The ADS1013/4/5 are fabricated in a small-geometry, low-voltage process. The analog inputs feature protection diodes to the supply rails. However, the current-handling ability of these diodes is limited, and the ADS1013/4/5 can be permanently damaged by analog input voltages that remain more than approximately 300mV beyond the rails for extended periods. One way to protect against overvoltage is to place current-limiting resistors on the input lines. The ADS1013/4/5 analog inputs can withstand momentary currents as large as 100mA.

If the ADS1013/4/5 are driven by an op amp with high-voltage supplies, such as ±12V, protection should be provided, even if the op amp is configured so that it does not output out-of-range voltages. Many op amps drift to one of the supply rails immediately when power is applied, usually before the input has stabilized; this momentary spike can damage the ADS1013/4/5. This incremental damage results in slow, long-term failure, which can be disastrous for permanently installed, low-maintenance systems.

If an op amp or other front-end circuitry is used with an ADS1013/4/5, performance characteristics must be taken into account when designing the application.

Copyright © 2009, Texas Instruments Incorporated

REVISION HISTORY

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision B (September 2009) to Revision C	Page
• Deleted operating temperature bullet from Features section	1
• Deleted *operating temperature range* parameter from Absolute Maximum Ratings table	2
• Deleted *Operating temperature* parameter from Temperature section of Electrical Characteristics table	4
• Changed Figure 2 to reflect maximum operating temperature	6
• Changed Figure 3 to reflect maximum operating temperature	6
• Changed Figure 4 to reflect maximum operating temperature	6
• Changed Figure 5 to reflect maximum operating temperature	6
• Changed Figure 6 to reflect maximum operating temperature	6

Copyright © 2009, Texas Instruments Incorporated

PACKAGING INFORMATION

Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/ Ball Finish	MSL Peak Temp [3]	Samples (Requires Login)
ADS1013IDGSR	ACTIVE	MSOP	DGS	10	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Purchase Samples
ADS1013IDGST	ACTIVE	MSOP	DGS	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Request Free Samples
ADS1013IRUGR	ACTIVE	X2QFN	RUG	10	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Purchase Samples
ADS1013IRUGT	ACTIVE	X2QFN	RUG	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Request Free Samples
ADS1014IDGSR	ACTIVE	MSOP	DGS	10	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Purchase Samples
ADS1014IDGST	ACTIVE	MSOP	DGS	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Request Free Samples
ADS1014IRUGR	ACTIVE	X2QFN	RUG	10	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Purchase Samples
ADS1014IRUGT	ACTIVE	X2QFN	RUG	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Request Free Samples
ADS1015IDGSR	ACTIVE	MSOP	DGS	10	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Purchase Samples
ADS1015IDGST	ACTIVE	MSOP	DGS	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	Request Free Samples
ADS1015IRUGR	ACTIVE	X2QFN	RUG	10	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Purchase Samples
ADS1015IRUGT	ACTIVE	X2QFN	RUG	10	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	Request Free Samples

[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.

Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

OTHER QUALIFIED VERSIONS OF ADS1015 :

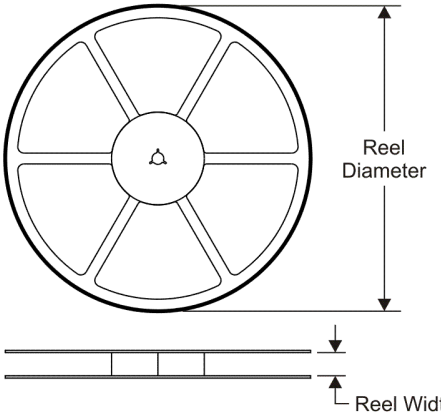
- Automotive: ADS1015-Q1

NOTE: Qualified Version Definitions:

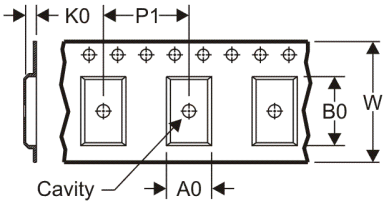
- Automotive - Q100 devices qualified for high-reliability automotive applications targeting zero defects

TAPE AND REEL INFORMATION

REEL DIMENSIONS

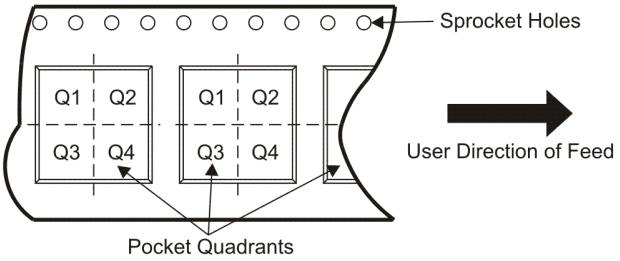


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

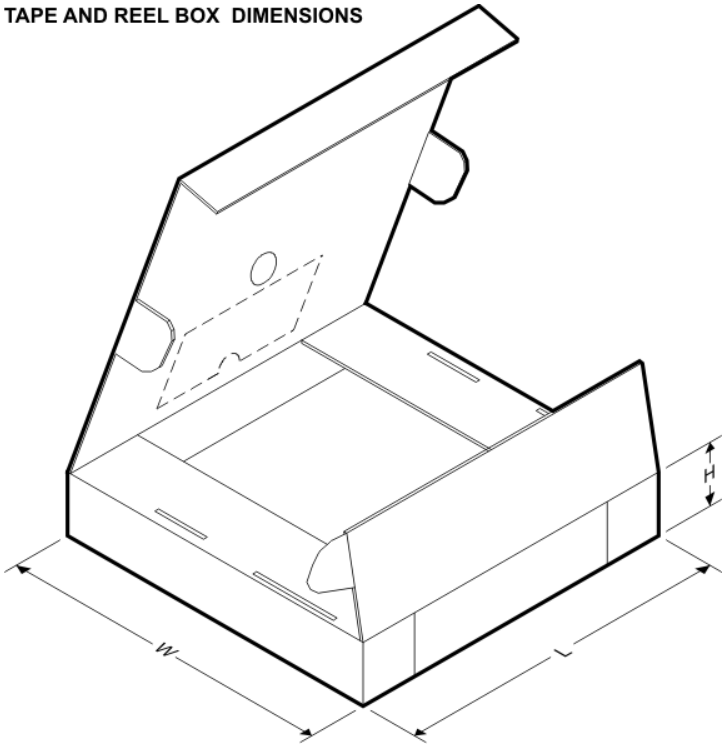
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
ADS1013IDGSR	MSOP	DGS	10	2500	330.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1013IDGST	MSOP	DGS	10	250	180.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1013IRUGR	X2QFN	RUG	10	3000	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1
ADS1013IRUGT	X2QFN	RUG	10	250	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1
ADS1014IDGSR	MSOP	DGS	10	2500	330.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1014IDGST	MSOP	DGS	10	250	180.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1014IRUGR	X2QFN	RUG	10	3000	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1
ADS1014IRUGT	X2QFN	RUG	10	250	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1
ADS1015IDGSR	MSOP	DGS	10	2500	330.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1015IDGST	MSOP	DGS	10	250	180.0	12.4	5.3	3.3	1.3	8.0	12.0	Q1
ADS1015IRUGR	X2QFN	RUG	10	3000	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1
ADS1015IRUGT	X2QFN	RUG	10	250	179.0	8.4	1.75	2.25	0.65	4.0	8.0	Q1

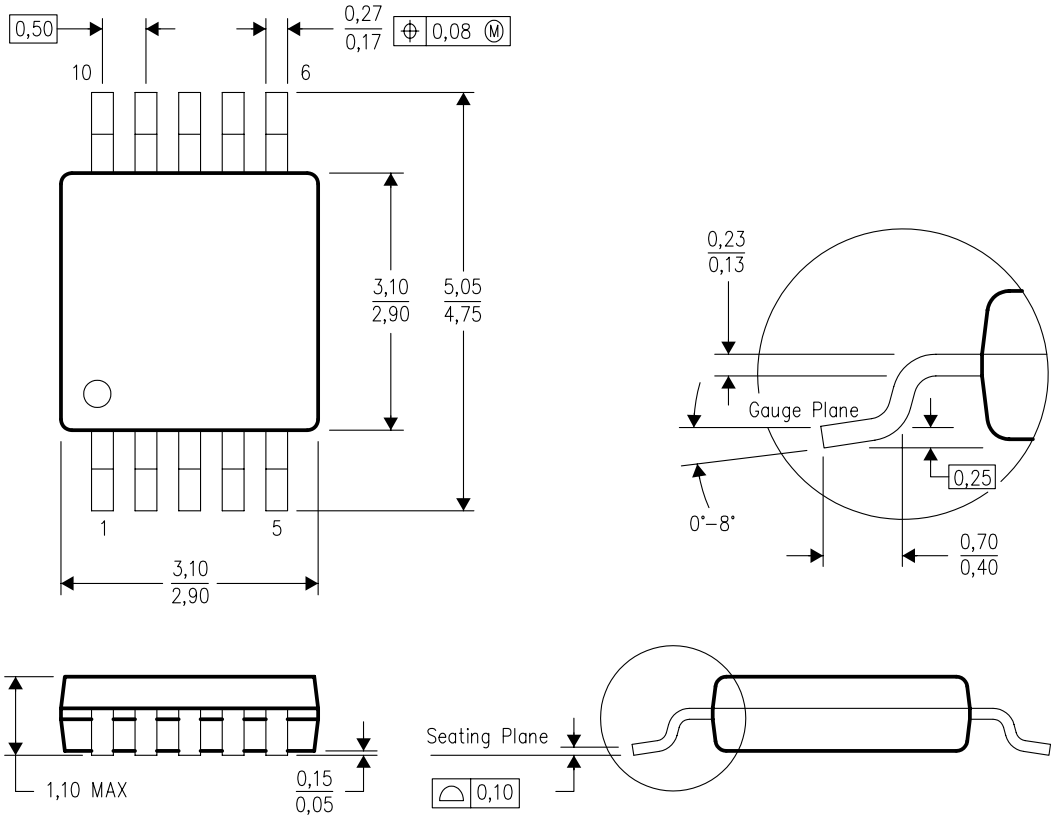
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
ADS1013IDGSR	MSOP	DGS	10	2500	370.0	355.0	55.0
ADS1013IDGST	MSOP	DGS	10	250	195.0	200.0	45.0
ADS1013IRUGR	X2QFN	RUG	10	3000	203.0	203.0	35.0
ADS1013IRUGT	X2QFN	RUG	10	250	203.0	203.0	35.0
ADS1014IDGSR	MSOP	DGS	10	2500	370.0	355.0	55.0
ADS1014IDGST	MSOP	DGS	10	250	195.0	200.0	45.0
ADS1014IRUGR	X2QFN	RUG	10	3000	203.0	203.0	35.0
ADS1014IRUGT	X2QFN	RUG	10	250	203.0	203.0	35.0
ADS1015IDGSR	MSOP	DGS	10	2500	370.0	355.0	55.0
ADS1015IDGST	MSOP	DGS	10	250	195.0	200.0	45.0
ADS1015IRUGR	X2QFN	RUG	10	3000	203.0	203.0	35.0
ADS1015IRUGT	X2QFN	RUG	10	250	203.0	203.0	35.0

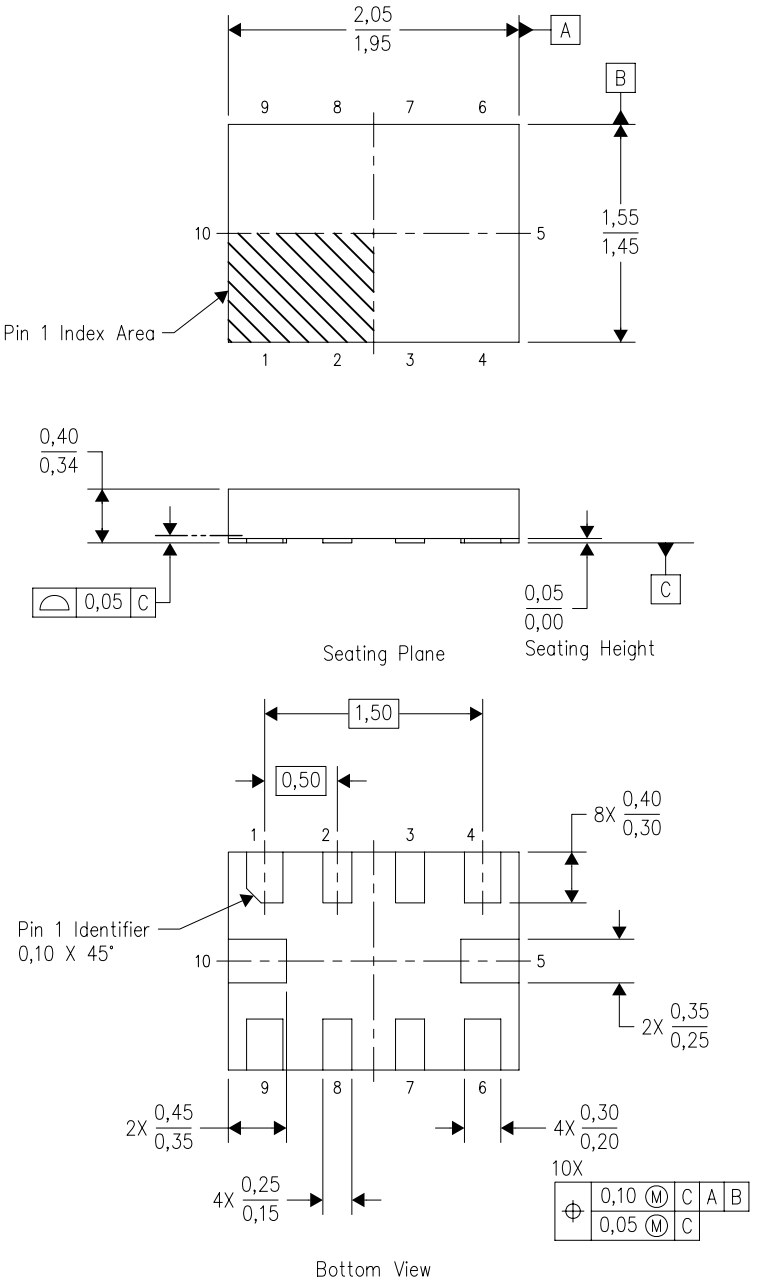
DGS (S–PDSO–G10) PLASTIC SMALL–OUTLINE PACKAGE



4073272/C 02/04

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion.
D. Falls within JEDEC MO–187 variation BA.

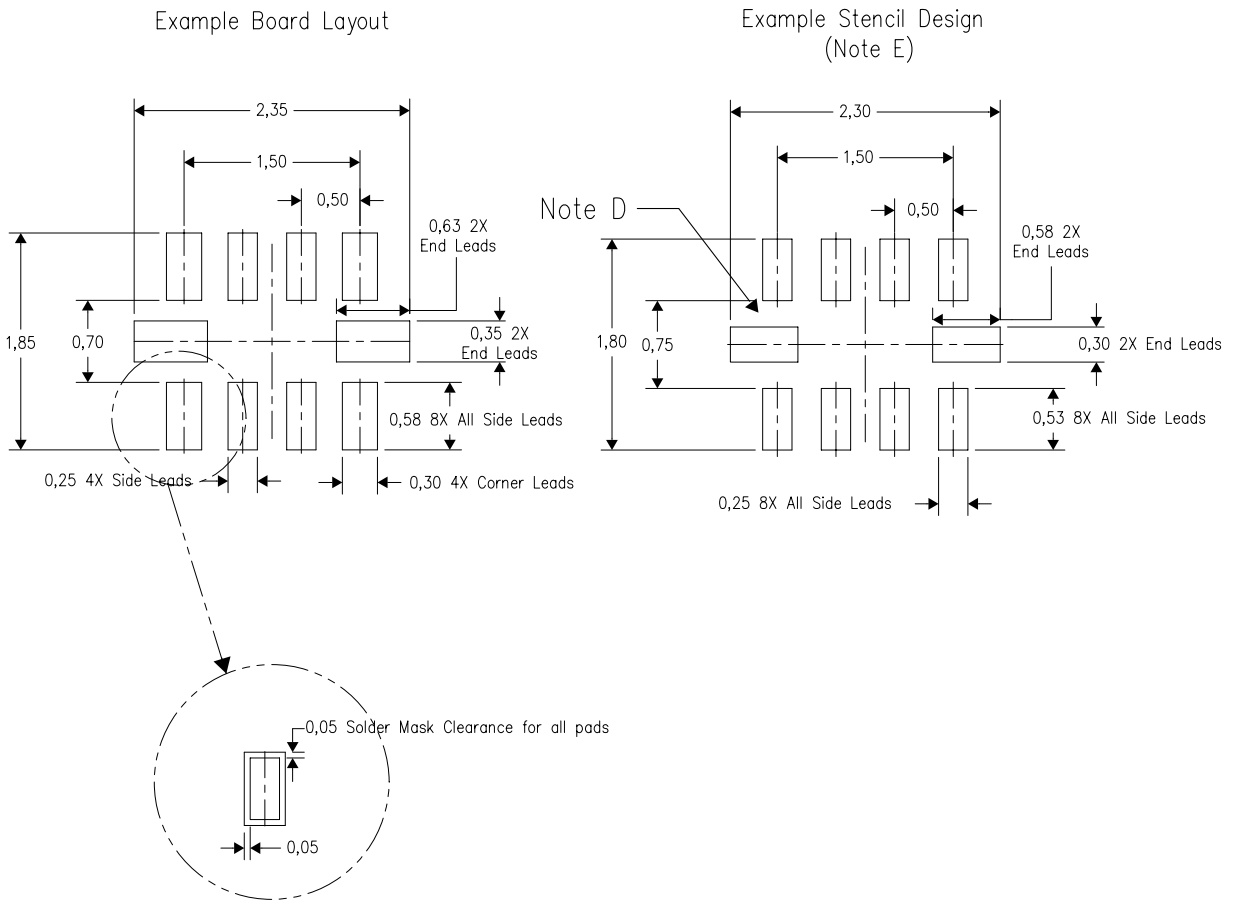
RUG (R–PQFP–N10) PLASTIC QUAD FLATPACK



4208528-3/B 04/2008

NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M–1994.
B. This drawing is subject to change without notice.
C. QFN (Quad Flatpack No–Lead) package configuration.
D. This package complies to JEDEC MO–288 variation X2EFD.

RUG (R–PQFP–N10)



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Customers should contact their board fabrication site for minimum solder mask web tolerances between signal pads.
E. Maximum stencil thickness 0,127 mm (5 mils). All linear dimensions are in millimeters.
F. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC 7525 for stencil design considerations.
G. Side aperture dimensions over–print land for acceptable area ratio > 0.66. Customer may reduce side aperture dimensions if stencil manufacturing process allows for sufficient release at smaller opening.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2010, Texas Instruments Incorporated

F MATLAB Code

F.1 Angle

```
%% Angle Calculation (Second Degree Polynomial):
%
% *Name:*        Carlos Rivera
%
% *Date:*        3/3/17
%
% *Objective:*
% Get data trends and polynomial for angle calculations
%
%% Initialize MATLAB Environment
%
% Before using this m file, we must clear the environment to make sure that
% all unnecessary previous definitions are deleted.

clear; clc; clf; cla; close all;
format long; format compact;
clc; clear;
close all;
format short;

%% Data Point Collection
% Data points collected for in 10 degree increments from 0 to 120 degrees.

MAX_POLYNOMIAL_POWER = 2;

% 0 degrees
D00 = [1.178; 1.155; 1.173; 1.166; 1.22];

% 10 degrees
D10 = [1.137; 1.143; 1.142; 1.140; 1.102];

% 20 degrees
D20 = [1.18; 1.147; 1.133; 1.114; 1.132];

% 30 degrees
D30 = [1.109; 1.101; 1.1072; 1.096; 1.095];

% 40 degrees
D40 = [1.058; 1.049; 1.065; 1.058; 1.058];

% 50 degrees
D50 = [0.9366; 0.9563; 0.9716; 0.9754; 0.9767];
```

```
% 60 degrees
D60 = [0.9448; 0.9607; 0.9521; 0.9552; 0.9588];

% 70 degrees
D70 = [0.8689; 0.8849; 0.8556; 0.8450; 0.8714];

% 80 degrees
D80 = [0.7674; 0.7887; 0.7753; 0.7586; 0.7703];

% 90 degrees
D90 = [0.5156; 0.4884; 0.4116; 0.4182; 0.4529];

% 100 degrees
D100 = [0.56; 0.55; 0.55; 0.57; 0.58];

% 110 degrees
D110 = [0.4474; 0.51; 0.50; 0.50; 0.57];

% 120 degrees
D120 = [0.58; 0.55; 0.55; 0.57; 0.57];

%% Average Calculation
%

D00_avg = mean(D00);
D10_avg = mean(D10);
D20_avg = mean(D20);
D30_avg = mean(D30);
D40_avg = mean(D40);
D50_avg = mean(D50);
D60_avg = mean(D60);
D70_avg = mean(D70);
D80_avg = mean(D80);
D90_avg = mean(D90);
D100_avg = mean(D100);
D110_avg = mean(D110);
D120_avg = mean(D120);

%% Plot Data and Calculate Polynomial
%

degrees = 0:10:120;
voltage = [D00_avg D10_avg D20_avg D30_avg ...
```

```
    D40_avg D50_avg D60_avg D70_avg ...
    D80_avg D90_avg D100_avg D110_avg D120_avg];

% plot(degrees,voltage,'o')
% title('Plot of y Versus t')

polynomial = polyfit(degrees, voltage, MAX_POLYNOMIAL_POWER);

t2 = 0:0.1:2.8;
polyline = polyval(polynomial,degrees);
figure
plot(degrees,voltage,'o',degrees,polyline)
title('Plot of Voltage vs Degrees')
xlabel('Degrees') % x-axis label
ylabel('Voltage') % y-axis label

fprintf('Polynomial: (%12.11f)x^2 + (%12.11f)x + (%12.11f)\n', ...
    polynomial(1,1), polynomial(1,2), polynomial(1,3))
```

F.2 Distance

```
%% Distance Calculation (Second Degree Polynomial):
%
% *Name:*        Carlos Rivera
%
% *Date:*        3/3/17
%
% *Objective:*
% Get data trends and polynomial for angle calculations
%
%% Initialize MATLAB Environment
%
% Before using this m file, we must clear the environment to make sure that
% all unnecessary previous definitions are deleted.

clear; clc; clf; cla; close all;
format long; format compact;
clc; clear;
close all;
format short;

%% Data Point Collection
% Data points collected for in 10 degree increments from 0 to 120 degrees.

MAX_POLYNOMIAL_POWER = 2;

% 5cm - STEPS OF 5cm
D005 = [0.527];

% 10cm
D010 = [0.603];

% 15cm
D015 = [0.681];

% 20cm
D020 = [0.736];

% 25cm - END STEPS OF 5cm
D025 = [0.765];

% 30cm
D030 = [0.819];
```

```
% 35cm
D035 = [0.817];

% 40cm
D040 = [0.817];

% 50cm
D050 = [0.840];

% 60cm
D060 = [0.841];

% 70cm
D070 = [0.889];

% 80cm
D080 = [0.910];

% 90cm
D090 = [0.915];

% 100cm
D100 = [0.936];

% 110cm
D110 = [0.939];

% 120cm
D120 = [0.960];

% 130cm
D130 = [0.986];

% 140cm
D140 = [0.957];

% 150cm
D150 = [1.032];

% 160cm
D160 = [1.101];

% 170cm
```

```
D170 = [1.108];

% 180cm
D180 = [1.120];

% 190cm
D190 = [1.140];

% 200cm
D200= [1.141];

% 210cm
D210 = [1.135];

% 230cm
D230 = [1.159];

% 250cm
D250 = [1.230];

% 260cm
D260 = [1.240];

% 280cm
D280 = [1.268];

% 300cm
D300 = [1.306];

% 470cm
D470 = [1.450];

%% Plot Data and Calculate Polynomial
%

voltage = [5 10 15 20 25 30 35 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190
200 ...
   210 230 250 260 280 300 470];
xmat = [D005 D010 D015 D020 D025 D030 D035 D040 D050 D060 D070 D080 D090 ...
   D100 D110 D120 D130 D140 D150 D160 D170 D180 D190 D200 D210 D230 ...
   D250 D260 D280 D300 D470];

% plot(xmat,voltage,'o')
```

```
% title('Plot of y Versus t')

polynomial = polyfit(xmat, voltage, MAX_POLYNOMIAL_POWER);

%t2 = 0:0.1:2.8;
polyline = polyval(polynomial,xmat);
figure(1)
plot(xmat,voltage,'o',xmat,polyline)
title('Plot of Voltage vs Degrees')
xlabel('Voltage') % x-axis label
ylabel('Distance (cm)') % y-axis label

fprintf('Polynomial: (%12.11f)x^2 + (%12.11f)x + (%12.11f)\n', ...
    polynomial(1,1), polynomial(1,2), polynomial(1,3))
```

G Transmitter Arduino Code

```
#include <SPI.h>
#include <RH_RF95.h>

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

// Change to 434.0 or other frequency, must match RX's freq!
#define RF95_FREQ 915.0

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);

bool TX_init_power = true;   //Have an initial power level or not, if not, TX power = 13dBm
(default)
int TX_inti_powerlvl = 10;        //Initialize a TX power level at the setup()

char Serial_Input;
int Serial_in_power;
bool TX_power = false;
int TX_powerlvl = 10;

void setup()
{
  pinMode(RFM95_RST, OUTPUT);
  digitalWrite(RFM95_RST, HIGH);

  while (!Serial);
  Serial.begin(9600);
  delay(100);

  Serial.println("Arduino LoRa TX Test!");

  // manual reset
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
    while (1);
  }
```

```
  Serial.println("LoRa radio init OK!");

  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }
  Serial.print("Set Freq to: "); Serial.println(RF95_FREQ);

  rf95.setTxPower(TX_inti_powerlvl, TX_init_power);        //Initialize TX power level at
intial_power

  if (TX_init_power == true) {
    Serial.print("TX Power Initialized to: ");
    Serial.print(TX_inti_powerlvl);
    Serial.println("dBm");
  }
  else {
    Serial.println("TX Power Initialized to: 13dBm");
  }


  // Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol,
CRC on
  // The default transmitter power is 13dBm, using PA_BOOST.
  // If you are using RFM95/96/97/98 modules which uses the PA_BOOST transmitter pin, then
  // you can set transmitter powers from 5 to 23 dBm:
}

int16_t packetnum = 0;  // packet counter, we increment per xmission

void loop()
{
  //Serial.println("Sending to rf95_server");
  // Send a message to rf95_server

  //Serial.print("Sending "); Serial.println(radiopacket);

  //  if (TX_power == true) {
  //    Serial.print("TX Power = ");
  //    Serial.print(TX_powerlvl);
  //    Serial.println("dBm");
  //  }
```

```
//
//  if (TX_power == false && TX_init_power == true) {
//    Serial.print("TX Power = ");
//    Serial.print(TX_inti_powerlvl);
//    Serial.println("dBm");
//  }
//
//  if (TX_power == false && TX_init_power == false) {
//    Serial.println("TX Power = 13dBm");
//  }
//
//  Serial.println("Sending..."); //delay(10);


//data being sent must be int type, other types must be converted
//last number = size of data in bytes

int radiopacket = 305;        //content of the packet being transmitted, as an int
rf95.send((uint8_t *)radiopacket, 1);
//rf95.waitPacketSent();


//Change power level using serial input
if (Serial.available() > 0)
{
  Serial_Input = Serial.read();

  if (Serial_Input == 'p')
  {
    TX_power = true;
    TX_powerlvl = setpowerlvl();
  }
  else
  {
    Serial.println("Invaild serial input");
  }
}

rf95.setTxPower(TX_powerlvl, false);

//Serial.println("Waiting for packet to complete..."); delay(10);

//delay(1000);
```

```
}


int setpowerlvl()
{
  int set_powerlvl;

  Serial.println("Power Level:");

  while (!Serial.available()) {
    delay(0);
  }
  if (Serial.available())
  {
    int temp_powerlvl = Serial.parseInt();
    if (temp_powerlvl > 23 || temp_powerlvl < 5)
    {
      Serial.println("Input power level out of range");
      set_powerlvl = TX_inti_powerlvl;
      //delay(400);
    }
    else
    {
      set_powerlvl = temp_powerlvl;
    }

  }

  return set_powerlvl;
}
```

H Raspberry Pi Code

The following files are the files needed in order to successfully run the software application on a chosen device, in this case a Raspberry Pi. The code is meant to be primarily run on a Raspberry Pi 3 with the following already installed:

- PyQt5
- NumPy

Below are all the files needed to run the software application

- application.py
- pyrebase.py
- globals.py
- threadpackage.py
- plot.py
- phasedetector.py

It should be noted that pyrebase.py was taken from the Pyrebase GitHub repository. Manually installing the library and calling it from the required files brought up errors. In order to mitigate this issue, the pyrebase.py file was included in the project and the needed classes and methods were referenced directly. This led to pyrebase.py becoming an integral part of the application. Credit for the code goes to GitHub user, thisbejim, for making the Python wrapper for the Firebase API.

```
 1 #!/usr/bin/python3
 2 # -*- coding: utf-8 -*-
 3
 4 import sys
 5 import time
 6 import phasedetector
 7 import globals
 8 import re
 9 import socket
10
11 from PyQt5.QtCore import Qt, QTimer, QThread
12 from PyQt5.QtWidgets import (QMainWindow, QTextEdit,
   QAction,
13                              QApplication, QGridLayout,
   QWidget,
14                              QLabel, QFrame, QSlider,
15                              QDialog, QLineEdit,
   QPushButton, QVBoxLayout, QMessageBox,
16                              QStackedLayout)
17 from PyQt5 import QtGui
18 from plot import Plot
19 from phasedetector import PhaseDetector
20 from threadpackage import ADCThread
21 from threadpackage import FilterThread
22 from threadpackage import DatabaseThread
23 import pyrebase
24
25 REMOTE_SERVER = "www.google.com"
26
27
28 # http://stackoverflow.com/questions/11812000/login-dialog-
   pyqt
29 class Login(QDialog):
30     def __init__(self, parent=None):
31         super(Login, self).__init__(parent)
32         self.textEmail = QLineEdit(self)
33         self.textPass = QLineEdit(self)
34         self.textPass.setEchoMode(QLineEdit.Password)
35         self.buttonLogin = QPushButton('Login', self)
36         self.buttonLogin.clicked.connect(self.handle_login)
37         username_label = QLabel("Email:")
38         password_label = QLabel("Password:")
39         layout = QGridLayout(self)
40         layout.addWidget(username_label, 0, 0)
41         layout.addWidget(password_label, 1, 0)
42         layout.addWidget(self.textEmail, 0, 1)
43         layout.addWidget(self.textPass, 1, 1)
44         layout.addWidget(self.buttonLogin, 2, 1)
45         self.setWindowTitle('RF Tracker')
46         self.auth = globals.firebase.auth()
```

```
47          self.EMAIL_REGEX = re.compile(r"(^[a-zA-Z0-9_
.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)")
48
49      def handle_login(self):
50          if self.EMAIL_REGEX.match(self.textEmail.text()):
51              user = self.auth.
sign_in_with_email_and_password(self.textEmail.text(), self
.textPass.text())
52              try:
53                  if user['idToken']:
54                      globals.user_token = user['idToken']
55                      self.accept()
56                  else:
57                      QMessageBox.warning(self, 'Error', 'Bad
 user or password!')
58              except KeyError:
59                  QMessageBox.warning(self, 'Error', 'Bad
user or password!')
60          else:
61              QMessageBox.warning(self, 'Error', 'Not a valid
 email address!')
62
63
64  class Settings(QDialog):
65      def __init__(self, parent=None):
66          super(Settings, self).__init__(parent)
67          self.line_color_text_field = QLineEdit(self)
68          self.point_color_text_field = QLineEdit(self)
69          self.buttonLogin = QPushButton('Accept', self)
70          self.buttonLogin.clicked.connect(self.update_values
)
71          line_label = QLabel("Plot Color")
72          point_label = QLabel("Point Color")
73          layout = QGridLayout(self)
74
75          layout.addWidget(line_label, 0, 0)
76          layout.addWidget(point_label, 1, 0)
77          layout.addWidget(self.line_color_text_field, 0, 1)
78          layout.addWidget(self.point_color_text_field, 1, 1)
79          layout.addWidget(self.buttonLogin, 2, 1)
80
81          self.setWindowTitle('Settings')
82          self.pattern = re.compile("#[0-9|A-F|a-f]"
83                                    "[0-9|A-F|a-f]"
84                                    "[0-9|A-F|a-f]"
85                                    "[0-9|A-F|a-f]"
86                                    "[0-9|A-F|a-f]"
87                                    "[0-9|A-F|a-f]")
88
89      def populate_text_fields(self, plot_line_color,
```

```
 89 plot_point_color):
 90         self.line_color_text_field.setText(plot_line_color
    )
 91         self.point_color_text_field.setText(
    plot_point_color)
 92 
 93     def update_values(self):
 94         if self.pattern.match(self.line_color_text_field.
    text()) and \
 95                 self.pattern.match(self.
    point_color_text_field.text()):
 96             self.line = self.line_color_text_field.text()
 97             self.point = self.point_color_text_field.text(
    )
 98             self.accept()
 99         else:
100             QMessageBox.warning(self, 'Error', 'Invalid
    Values')
101 
102 
103 class RFTracker(QMainWindow):
104     def __init__(self):
105         self.current_page = 0
106         self.MAX_ACCURATE_RANGE = 2047
107         self.MIN_ACCURATE_RANGE = 50
108         super().__init__()
109         self.initialize_ui()
110 
111     def initialize_ui(self):
112         self.create_home_widget()
113         self.create_menubars()
114 
115         # Begin QTimer Poll and Read ADS
116         timer = QTimer(self)
117         timer.timeout.connect(self.plot.nextAnimationFrame
    )
118         timer.start(10)
119 
120         # TODO: test lines
121         test_layout = QStackedLayout()
122         test_layout.addWidget(self.home_widget)
123         # test_layout.setCurrentIndex(0)
124 
125         self.central_widget = QWidget()
126         self.central_widget.setLayout(test_layout)
127         self.setCentralWidget(self.central_widget)
128 
129         self.showFullScreen()
130         self.setWindowTitle('RF Tracker')
131         self.show()
```

```
132
133     def settings(self):
134         settings = Settings()
135         settings.populate_text_fields(self.plot.line_color
    , self.plot.point_color)
136         settings.exec_()
137         self.plot.change_colors(settings.line, settings.
    point)
138
139     def resume_tracking(self):
140         self.plot.continue_tracking()
141
142     def pause_tracking(self):
143         self.plot.pause_tracking()
144
145     def refresh_grid(self):
146         self.plot.refresh()
147
148     def create_menubars(self):
149         # Actions
150         exit_action = QAction(QtGui.QIcon('icons/exit.png'
    ), 'Exit', self)
151         exit_action.setShortcut('Ctrl+Q')
152         exit_action.setStatusTip('Exit Application')
153         exit_action.triggered.connect(self.close)
154
155         refresh_action = QAction(QtGui.QIcon('icons/
    refresh.png'), 'Refresh Grid', self)
156         refresh_action.setShortcut('Ctrl+R')
157         refresh_action.setStatusTip('Refresh Grid')
158         refresh_action.triggered.connect(self.refresh_grid
    )
159
160         settings_action = QAction(QtGui.QIcon('icons/
    adjustSettings.png'), 'Settings', self)
161         settings_action.setShortcut('Ctrl+S')
162         settings_action.setStatusTip('Adjust Settings')
163         settings_action.triggered.connect(self.settings)
164
165         play_action = QAction(QtGui.QIcon('icons/play.png'
    ), 'Start Tracking', self)
166         play_action.setShortcut('Ctrl+P')
167         play_action.setStatusTip('Continue Tracking')
168         play_action.triggered.connect(self.resume_tracking
    )
169
170         pause_action = QAction(QtGui.QIcon('icons/pause.
    png'), 'Pause Tracking', self)
171         pause_action.setShortcut('Ctrl+O')
172         pause_action.setStatusTip('Pause Tracking')
```

```
173         pause_action.triggered.connect(self.pause_tracking
    )
174
175         # Status Bar
176         self.statusBar().showMessage('Ready')
177
178         # Menu Bar
179         menubar = self.menuBar()
180
181         # File Menu
182         file_menu = menubar.addMenu('&File')
183         file_menu.addAction(exit_action)
184
185         # Grid Menu and actions
186         grid_menu = menubar.addMenu('&Grid')
187         grid_menu.addAction(play_action)
188         grid_menu.addAction(pause_action)
189
190         # Edit Menu
191         edit_menu = menubar.addMenu('&Edit')
192         edit_menu.addAction(refresh_action)
193         edit_menu.addAction(settings_action)
194
195         # Tool Bar
196         toolbar = self.addToolBar('Exit')
197         toolbar.addAction(exit_action)
198         toolbar.addAction(refresh_action)
199         toolbar.addAction(settings_action)
200         toolbar.addAction(pause_action)
201         toolbar.addAction(play_action)
202
203     def create_home_widget(self):
204         # fromRow, fromColumn, rowSpan, columnSpan
205         self.home_widget = QWidget()
206         layout = QGridLayout()
207         self.home_widget.setLayout(layout)
208
209         label = self.create_label(text="Distance")
210         label2 = self.create_label(text="Sector A")
211         label3 = self.create_label(text="Angle")
212         layout.addWidget(label3, 1, 1)
213         layout.addWidget(label2, 0, 0, 2, 1)
214         layout.addWidget(label, 0, 1)
215
216         # Make Plot
217         self.plot = Plot()
218         layout.addWidget(self.plot, 0, 2, 2, 1)
219         # slider = QSlider(Qt.Vertical, self)
220         # slider.setStatusTip('Zoom')
221         # layout.addWidget(slider, 0, 3)
```

```
222
223     def create_label(self, text):
224         label = QLabel(text)
225         label.setAlignment(Qt.AlignCenter)
226         label.setMargin(2)
227         label.setFrameStyle(QFrame.Box | QFrame.Sunken)
228         label.setMinimumSize(self.sizeHint())
229         return label
230
231
232 def is_connected():
233     try:
234         # see if we can resolve the host name -- tells us
    if there is
235         # a DNS listening
236         host = socket.gethostbyname(REMOTE_SERVER)
237         # connect to the host -- tells us if the host is
    reachable
238         temp = socket.create_connection((host, 80), 2)
239         return True
240     except:
241         pass
242     return False
243
244
245 def setup_firebase():
246     config = {
247         "apiKey": "AIzaSyCobF9FiE7NMo6RUISeEcTWQb9qmL2MukU
    ",
248         "authDomain": "rf-tracker.firebaseapp.com",
249         "databaseURL": "https://rf-tracker.firebaseio.com"
    ,
250         "storageBucket": "rf-tracker.appspot.com",
251     }
252     firebase = pyrebase.initialize_app(config)
253     return firebase
254
255 if __name__ == '__main__':
256     # initialize global variables
257     globals.init()
258
259     if is_connected():
260         # setup firebase
261         globals.firebase = setup_firebase()
262         globals.database = globals.firebase.database()
263
264         # start application
265         app = QApplication(sys.argv)
266         login = Login()
267
```

```
268         if login.exec_() == QDialog.Accepted:
269             ex = RFTracker()
270 
271             # ADC Read Thread
272             threadADC = ADCThread()
273             threadADC.finished.connect(app.exit)
274             threadADC.start()
275 
276             # Filter Thread
277             threadFilter = FilterThread()
278             threadFilter.finished.connect(app.exit)
279             threadFilter.start()
280 
281             # Database Thread
282             databaseThread = DatabaseThread(login.
    textEmail.text())
283             databaseThread.finished.connect(app.exit)
284             databaseThread.start()
285 
286             sys.exit(app.exec_())
287     else:
288         app = QApplication(sys.argv)
289         ex = RFTracker()
290 
291         # ADC Read Thread
292         threadADC = ADCThread()
293         threadADC.finished.connect(app.exit)
294         threadADC.start()
295 
296         # Filter Thread
297         threadFilter = FilterThread()
298         threadFilter.finished.connect(app.exit)
299         threadFilter.start()
300 
301         sys.exit(app.exec_())
302 
```

```
 1 import requests
 2 from requests import Session
 3 from requests.exceptions import HTTPError
 4
 5 try:
 6     from urllib.parse import urlencode, quote
 7 except:
 8     from urllib import urlencode, quote
 9
10 import json
11 import math
12 from random import uniform
13 import time
14 from collections import OrderedDict
15 from sseclient import SSEClient
16 import threading
17 import socket
18 from oauth2client.service_account import
   ServiceAccountCredentials
19 from gcloud import storage
20 # from requests.packages.urllib3.contrib.appengine import
   is_appengine_sandbox
21 # from requests_toolbelt.adapters import appengine
22
23 import python_jwt as jwt
24 from Crypto.PublicKey import RSA
25 import datetime
26
27
28 def initialize_app(config):
29     return Firebase(config)
30
31
32 class Firebase:
33     """ Firebase Interface """
34     def __init__(self, config):
35         self.api_key = config["apiKey"]
36         self.auth_domain = config["authDomain"]
37         self.database_url = config["databaseURL"]
38         self.storage_bucket = config["storageBucket"]
39         self.credentials = None
40         self.requests = requests.Session()
41         if config.get("serviceAccount"):
42             scopes = [
43                 'https://www.googleapis.com/auth/firebase.
   database',
44                 'https://www.googleapis.com/auth/userinfo.
   email',
45                 "https://www.googleapis.com/auth/cloud-
   platform"
```

```
46              ]
47              service_account_type = type(config["
    serviceAccount"])
48              if service_account_type is str:
49                  self.credentials =
    ServiceAccountCredentials.from_json_keyfile_name(config["
    serviceAccount"], scopes)
50              if service_account_type is dict:
51                  self.credentials =
    ServiceAccountCredentials.from_json_keyfile_dict(config["
    serviceAccount"], scopes)
52          #if is_appengine_sandbox():
53          #    # Fix error in standard GAE environment
54          #    # is releated to https://github.com/
    kennethreitz/requests/issues/3187
55          #    # ProtocolError('Connection aborted.', error(
    13, 'Permission denied'))
56          #    adapter = appengine.AppEngineAdapter(
    max_retries=3)
57          #else:
58          #    adapter = requests.adapters.HTTPAdapter(
    max_retries=3)
59          adapter = requests.adapters.HTTPAdapter(max_retries
    =3)
60
61          for scheme in ('http://', 'https://'):
62              self.requests.mount(scheme, adapter)
63
64      def auth(self):
65          return Auth(self.api_key, self.requests, self.
    credentials)
66
67      def database(self):
68          return Database(self.credentials, self.api_key,
    self.database_url, self.requests)
69
70      def storage(self):
71          return Storage(self.credentials, self.
    storage_bucket, self.requests)
72
73
74  class Auth:
75      """ Authentication Service """
76      def __init__(self, api_key, requests, credentials):
77          self.api_key = api_key
78          self.current_user = None
79          self.requests = requests
80          self.credentials = credentials
81
82      def sign_in_with_email_and_password(self, email,
```

```
 82 password):
 83         request_ref = "https://www.googleapis.com/
    identitytoolkit/v3/relyingparty/verifyPassword?key={0}".
    format(self.api_key)
 84         headers = {"content-type": "application/json;
    charset=UTF-8"}
 85         data = json.dumps({"email": email, "password":
    password, "returnSecureToken": True})
 86         request_object = requests.post(request_ref,
    headers=headers, data=data)
 87         try:
 88             raise_detailed_error(request_object)
 89         except HTTPError:
 90             pass
 91         self.current_user = request_object.json()
 92         return request_object.json()
 93
 94     def create_custom_token(self, uid, additional_claims=
    None):
 95         service_account_email = self.credentials.
    service_account_email
 96         private_key = RSA.importKey(self.credentials.
    _private_key_pkcs8_pem)
 97         payload = {
 98             "iss": service_account_email,
 99             "sub": service_account_email,
100             "aud": "https://identitytoolkit.googleapis.com
    /google.identity.identitytoolkit.v1.IdentityToolkit",
101             "uid": uid
102         }
103         if additional_claims:
104             payload["claims"] = additional_claims
105         exp = datetime.timedelta(minutes=60)
106         return jwt.generate_jwt(payload, private_key, "
    RS256", exp)
107
108     def sign_in_with_custom_token(self, token):
109         request_ref = "https://www.googleapis.com/
    identitytoolkit/v3/relyingparty/verifyCustomToken?key={0}"
    .format(self.api_key)
110         headers = {"content-type": "application/json;
    charset=UTF-8"}
111         data = json.dumps({"returnSecureToken": True, "
    token": token})
112         request_object = requests.post(request_ref,
    headers=headers, data=data)
113         raise_detailed_error(request_object)
114         return request_object.json()
115
116     def refresh(self, refresh_token):
```

```
117         request_ref = "https://securetoken.googleapis.com/v1/token?key={0}".format(self.api_key)
118         headers = {"content-type": "application/json; charset=UTF-8"}
119         data = json.dumps({"grantType": "refresh_token", "refreshToken": refresh_token})
120         request_object = requests.post(request_ref, headers=headers, data=data)
121         raise_detailed_error(request_object)
122         request_object_json = request_object.json()
123         # handle weirdly formatted response
124         user = {
125             "userId": request_object_json["user_id"],
126             "idToken": request_object_json["id_token"],
127             "refreshToken": request_object_json["refresh_token"]
128         }
129         return user
130 
131     def get_account_info(self, id_token):
132         request_ref = "https://www.googleapis.com/identitytoolkit/v3/relyingparty/getAccountInfo?key={0}".format(self.api_key)
133         headers = {"content-type": "application/json; charset=UTF-8"}
134         data = json.dumps({"idToken": id_token})
135         request_object = requests.post(request_ref, headers=headers, data=data)
136         raise_detailed_error(request_object)
137         return request_object.json()
138 
139     def send_email_verification(self, id_token):
140         request_ref = "https://www.googleapis.com/identitytoolkit/v3/relyingparty/getOobConfirmationCode?key={0}".format(self.api_key)
141         headers = {"content-type": "application/json; charset=UTF-8"}
142         data = json.dumps({"requestType": "VERIFY_EMAIL", "idToken": id_token})
143         request_object = requests.post(request_ref, headers=headers, data=data)
144         raise_detailed_error(request_object)
145         return request_object.json()
146 
147     def send_password_reset_email(self, email):
148         request_ref = "https://www.googleapis.com/identitytoolkit/v3/relyingparty/getOobConfirmationCode?key={0}".format(self.api_key)
149         headers = {"content-type": "application/json; charset=UTF-8"}
```

```
150         data = json.dumps({"requestType": "PASSWORD_RESET"
    , "email": email})
151         request_object = requests.post(request_ref,
    headers=headers, data=data)
152         raise_detailed_error(request_object)
153         return request_object.json()
154 
155     def verify_password_reset_code(self, reset_code,
    new_password):
156         request_ref = "https://www.googleapis.com/
    identitytoolkit/v3/relyingparty/resetPassword?key={0}".
    format(self.api_key)
157         headers = {"content-type": "application/json;
    charset=UTF-8"}
158         data = json.dumps({"oobCode": reset_code, "
    newPassword": new_password})
159         request_object = requests.post(request_ref,
    headers=headers, data=data)
160         raise_detailed_error(request_object)
161         return request_object.json()
162 
163     def create_user_with_email_and_password(self, email,
    password):
164         request_ref = "https://www.googleapis.com/
    identitytoolkit/v3/relyingparty/signupNewUser?key={0}".
    format(self.api_key)
165         headers = {"content-type": "application/json;
    charset=UTF-8" }
166         data = json.dumps({"email": email, "password":
    password, "returnSecureToken": True})
167         request_object = requests.post(request_ref,
    headers=headers, data=data)
168         raise_detailed_error(request_object)
169         return request_object.json()
170 
171 
172 class Database:
173     """ Database Service """
174     def __init__(self, credentials, api_key, database_url,
     requests):
175 
176         if not database_url.endswith('/'):
177             url = ''.join([database_url, '/'])
178         else:
179             url = database_url
180 
181         self.credentials = credentials
182         self.api_key = api_key
183         self.database_url = url
184         self.requests = requests
```

```
185
186         self.path = ""
187         self.build_query = {}
188         self.last_push_time = 0
189         self.last_rand_chars = []
190
191     def order_by_key(self):
192         self.build_query["orderBy"] = "$key"
193         return self
194
195     def order_by_value(self):
196         self.build_query["orderBy"] = "$value"
197         return self
198
199     def order_by_child(self, order):
200         self.build_query["orderBy"] = order
201         return self
202
203     def start_at(self, start):
204         self.build_query["startAt"] = start
205         return self
206
207     def end_at(self, end):
208         self.build_query["endAt"] = end
209         return self
210
211     def equal_to(self, equal):
212         self.build_query["equalTo"] = equal
213         return self
214
215     def limit_to_first(self, limit_first):
216         self.build_query["limitToFirst"] = limit_first
217         return self
218
219     def limit_to_last(self, limit_last):
220         self.build_query["limitToLast"] = limit_last
221         return self
222
223     def shallow(self):
224         self.build_query["shallow"] = True
225         return self
226
227     def child(self, *args):
228         new_path = "/".join([str(arg) for arg in args])
229         if self.path:
230             self.path += "/{}".format(new_path)
231         else:
232             if new_path.startswith("/"):
233                 new_path = new_path[1:]
234             self.path = new_path
```

```
235         return self
236 
237     def build_request_url(self, token):
238         parameters = {}
239         if token:
240             parameters['auth'] = token
241         for param in list(self.build_query):
242             if type(self.build_query[param]) is str:
243                 parameters[param] = quote('"' + self.
    build_query[param] + '"')
244             elif type(self.build_query[param]) is bool:
245                 parameters[param] = "true" if self.
    build_query[param] else "false"
246             else:
247                 parameters[param] = self.build_query[param
    ]
248         # reset path and build_query for next query
249         request_ref = '{0}{1}.json?{2}'.format(self.
    database_url, self.path, urlencode(parameters))
250         self.path = ""
251         self.build_query = {}
252         return request_ref
253 
254     def build_headers(self, token=None):
255         headers = {"content-type": "application/json;
    charset=UTF-8"}
256         if not token and self.credentials:
257             access_token = self.credentials.
    get_access_token().access_token
258             headers['Authorization'] = 'Bearer ' +
    access_token
259         return headers
260 
261     def get(self, token=None, json_kwargs={}):
262         build_query = self.build_query
263         query_key = self.path.split("/")[-1]
264         request_ref = self.build_request_url(token)
265         # headers
266         headers = self.build_headers(token)
267         # do request
268         request_object = self.requests.get(request_ref,
    headers=headers)
269         raise_detailed_error(request_object)
270         request_dict = request_object.json(**json_kwargs)
271 
272         # if primitive or simple query return
273         if isinstance(request_dict, list):
274             return PyreResponse(convert_list_to_pyre(
    request_dict), query_key)
275         if not isinstance(request_dict, dict):
```

```
276             return PyreResponse(request_dict, query_key)
277         if not build_query:
278             return PyreResponse(convert_to_pyre(
    request_dict.items()), query_key)
279         # return keys if shallow
280         if build_query.get("shallow"):
281             return PyreResponse(request_dict.keys(),
    query_key)
282         # otherwise sort
283         sorted_response = None
284         if build_query.get("orderBy"):
285             if build_query["orderBy"] == "$key":
286                 sorted_response = sorted(request_dict.
    items(), key=lambda item: item[0])
287             elif build_query["orderBy"] == "$value":
288                 sorted_response = sorted(request_dict.
    items(), key=lambda item: item[1])
289             else:
290                 sorted_response = sorted(request_dict.
    items(), key=lambda item: item[1][build_query["orderBy"]])
291         return PyreResponse(convert_to_pyre(
    sorted_response), query_key)
292 
293     def push(self, data, token=None, json_kwargs={}):
294         request_ref = self.check_token(self.database_url,
    self.path, token)
295         self.path = ""
296         headers = self.build_headers(token)
297         request_object = self.requests.post(request_ref,
    headers=headers, data=json.dumps(data, **json_kwargs).
    encode("utf-8"))
298         raise_detailed_error(request_object)
299         return request_object.json()
300 
301     def set(self, data, token=None, json_kwargs={}):
302         request_ref = self.check_token(self.database_url,
    self.path, token)
303         self.path = ""
304         headers = self.build_headers(token)
305         request_object = self.requests.put(request_ref,
    headers=headers, data=json.dumps(data, **json_kwargs).
    encode("utf-8"))
306         raise_detailed_error(request_object)
307         return request_object.json()
308 
309     def update(self, data, token=None, json_kwargs={}):
310         request_ref = self.check_token(self.database_url,
    self.path, token)
311         self.path = ""
312         headers = self.build_headers(token)
```

```
313         request_object = self.requests.patch(request_ref,
    headers=headers, data=json.dumps(data, **json_kwargs).
    encode("utf-8"))
314         raise_detailed_error(request_object)
315         return request_object.json()
316 
317     def remove(self, token=None):
318         request_ref = self.check_token(self.database_url,
    self.path, token)
319         self.path = ""
320         headers = self.build_headers(token)
321         request_object = self.requests.delete(request_ref,
     headers=headers)
322         raise_detailed_error(request_object)
323         return request_object.json()
324 
325     def stream(self, stream_handler, token=None, stream_id
    =None):
326         request_ref = self.build_request_url(token)
327         return Stream(request_ref, stream_handler, self.
    build_headers, stream_id)
328 
329     def check_token(self, database_url, path, token):
330         if token:
331             return '{0}{1}.json?auth={2}'.format(
    database_url, path, token)
332         else:
333             return '{0}{1}.json'.format(database_url, path
    )
334 
335     def generate_key(self):
336         push_chars = '-
    0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstu
    vwxyz'
337         now = int(time.time() * 1000)
338         duplicate_time = now == self.last_push_time
339         self.last_push_time = now
340         time_stamp_chars = [0] * 8
341         for i in reversed(range(0, 8)):
342             time_stamp_chars[i] = push_chars[now % 64]
343             now = int(math.floor(now / 64))
344         new_id = "".join(time_stamp_chars)
345         if not duplicate_time:
346             for i in range(0, 12):
347                 self.last_rand_chars.append(int(math.floor
    (uniform(0, 1) * 64)))
348         else:
349             for i in range(0, 11):
350                 if self.last_rand_chars[i] == 63:
351                     self.last_rand_chars[i] = 0
```

```
352                 self.last_rand_chars[i] += 1
353         for i in range(0, 12):
354             new_id += push_chars[self.last_rand_chars[i]]
355         return new_id
356 
357     def sort(self, origin, by_key):
358         # unpack pyre objects
359         pyres = origin.each()
360         new_list = []
361         for pyre in pyres:
362             new_list.append(pyre.item)
363         # sort
364         data = sorted(dict(new_list).items(), key=lambda
    item: item[1][by_key])
365         return PyreResponse(convert_to_pyre(data), origin.
    key())
366 
367 
368 class Storage:
369     """ Storage Service """
370     def __init__(self, credentials, storage_bucket,
    requests):
371         self.storage_bucket = "https://firebasestorage.
    googleapis.com/v0/b/" + storage_bucket
372         self.credentials = credentials
373         self.requests = requests
374         self.path = ""
375         if credentials:
376             client = storage.Client(credentials=
    credentials, project=storage_bucket)
377             self.bucket = client.get_bucket(storage_bucket
    )
378 
379     def child(self, *args):
380         new_path = "/".join(args)
381         if self.path:
382             self.path += "/{}".format(new_path)
383         else:
384             if new_path.startswith("/"):
385                 new_path = new_path[1:]
386             self.path = new_path
387         return self
388 
389     def put(self, file, token=None):
390         # reset path
391         path = self.path
392         self.path = None
393         if isinstance(file, str):
394             file_object = open(file, 'rb')
395         else:
```

```
396             file_object = file
397         request_ref = self.storage_bucket + "/o?name={0}".
    format(path)
398         if token:
399             headers = {"Authorization": "Firebase " +
    token}
400             request_object = self.requests.post(
    request_ref, headers=headers, data=file_object)
401             raise_detailed_error(request_object)
402             return request_object.json()
403         elif self.credentials:
404             blob = self.bucket.blob(path)
405             if isinstance(file, str):
406                 return blob.upload_from_filename(filename=
    file)
407             else:
408                 return blob.upload_from_file(file_obj=file
    )
409         else:
410             request_object = self.requests.post(
    request_ref, data=file_object)
411             raise_detailed_error(request_object)
412             return request_object.json()
413 
414     def delete(self, name):
415         self.bucket.delete_blob(name)
416 
417     def download(self, filename, token=None):
418         # remove leading backlash
419         path = self.path
420         url = self.get_url(token)
421         self.path = None
422         if path.startswith('/'):
423             path = path[1:]
424         if self.credentials:
425             blob = self.bucket.get_blob(path)
426             blob.download_to_filename(filename)
427         else:
428             r = requests.get(url, stream=True)
429             if r.status_code == 200:
430                 with open(filename, 'wb') as f:
431                     for chunk in r:
432                         f.write(chunk)
433 
434     def get_url(self, token):
435         path = self.path
436         self.path = None
437         if path.startswith('/'):
438             path = path[1:]
439         if token:
```

```
440             return "{0}/o/{1}?alt=media&token={2}".format(
    self.storage_bucket, quote(path, safe=''), token)
441         return "{0}/o/{1}?alt=media".format(self.
    storage_bucket, quote(path, safe=''))
442 
443     def list_files(self):
444         return self.bucket.list_blobs()
445 
446 
447 def raise_detailed_error(request_object):
448     try:
449         request_object.raise_for_status()
450     except HTTPError as e:
451         # raise detailed error message
452         # TODO: Check if we get a { "error" : "Permission
    denied." } and handle automatically
453         raise HTTPError(e, request_object.text)
454 
455 
456 def convert_to_pyre(items):
457     pyre_list = []
458     for item in items:
459         pyre_list.append(Pyre(item))
460     return pyre_list
461 
462 
463 def convert_list_to_pyre(items):
464     pyre_list = []
465     for item in items:
466         pyre_list.append(Pyre([items.index(item), item]))
467     return pyre_list
468 
469 
470 class PyreResponse:
471     def __init__(self, pyres, query_key):
472         self.pyres = pyres
473         self.query_key = query_key
474 
475     def val(self):
476         if isinstance(self.pyres, list):
477             # unpack pyres into OrderedDict
478             pyre_list = []
479             # if firebase response was a list
480             if isinstance(self.pyres[0].key(), int):
481                 for pyre in self.pyres:
482                     pyre_list.append(pyre.val())
483                 return pyre_list
484             # if firebase response was a dict with keys
485             for pyre in self.pyres:
486                 pyre_list.append((pyre.key(), pyre.val()))
```

```
            return OrderedDict(pyre_list)
        else:
            # return primitive or simple query results
            return self.pyres

    def key(self):
        return self.query_key

    def each(self):
        if isinstance(self.pyres, list):
            return self.pyres


class Pyre:
    def __init__(self, item):
        self.item = item

    def val(self):
        return self.item[1]

    def key(self):
        return self.item[0]


class KeepAuthSession(Session):
    """
    A session that doesn't drop Authentication on redirects between domains.
    """

    def rebuild_auth(self, prepared_request, response):
        pass


class ClosableSSEClient(SSEClient):
    def __init__(self, *args, **kwargs):
        self.should_connect = True
        super(ClosableSSEClient, self).__init__(*args, **kwargs)

    def _connect(self):
        if self.should_connect:
            super(ClosableSSEClient, self)._connect()
        else:
            raise StopIteration()

    def close(self):
        self.should_connect = False
        self.retry = 0
        self.resp.raw._fp.fp.raw._sock.shutdown(socket.
```

```
534 SHUT_RDWR)
535         self.resp.raw._fp.fp.raw._sock.close()
536 
537 
538 class Stream:
539     def __init__(self, url, stream_handler, build_headers,
     stream_id):
540         self.build_headers = build_headers
541         self.url = url
542         self.stream_handler = stream_handler
543         self.stream_id = stream_id
544         self.sse = None
545         self.thread = None
546         self.start()
547 
548     def make_session(self):
549         """
550         Return a custom session object to be passed to the
     ClosableSSEClient.
551         """
552         session = KeepAuthSession()
553         return session
554 
555     def start(self):
556         self.thread = threading.Thread(target=self.
    start_stream)
557         self.thread.start()
558         return self
559 
560     def start_stream(self):
561         self.sse = ClosableSSEClient(self.url, session=
    self.make_session(), build_headers=self.build_headers)
562         for msg in self.sse:
563             if msg:
564                 msg_data = json.loads(msg.data)
565                 msg_data["event"] = msg.event
566                 if self.stream_id:
567                     msg_data["stream_id"] = self.stream_id
568                 self.stream_handler(msg_data)
569 
570     def close(self):
571         while not self.sse and not hasattr(self.sse, 'resp
    '):
572             time.sleep(0.001)
573         self.sse.running = False
574         self.sse.close()
575         self.thread.join()
576         return self
```

```
from PyQt5.QtCore import QMutex
from enum import Enum
from pyrebase import Firebase
import numpy as np


class Sector(Enum):
    A = 0
    B = 1
    C = 2


def init():
    global amplitudeA
    amplitudeA = 0
    global amplitudeB
    amplitudeB = 0
    global amplitudeC
    amplitudeC = 0
    global pAmplitudeA
    amplitudeA = 0
    global pAmplitudeB
    amplitudeB = 0
    global pAmplitudeC
    amplitudeC = 0
    global distance
    distance = 0
    global previous_error
    previous_error = 0

    global firebase
    firebase = False
    global database
    database = False

    global user_token
    user_token = ""

    # 0 -> amplitude A
    # 1 -> amplitude B
    # 2 -> amplitude C
    # 3 -> distance
    #n_timesteps = 1
    #n_dim_state = 5
    #global prev_state_means
    #prev_state_means = np.zeros((n_timesteps, n_dim_state
))
    #global prev_covariances
    #prev_covariances = np.zeros((n_timesteps, n_dim_state
, n_dim_state))
```

```
49      #global curr_state_means
50      #curr_state_means = np.zeros((n_timesteps, n_dim_state
    ))
51      #global curr_covariances
52      #curr_covariances = np.zeros((n_timesteps, n_dim_state
    , n_dim_state))
53
54      n_timesteps = 2
55      n_dim_state = 5
56      global filtered_state_means
57      filtered_state_means = np.zeros((n_timesteps,
    n_dim_state))
58      global filtered_state_covariances
59      filtered_state_covariances = np.zeros((n_timesteps,
    n_dim_state, n_dim_state))
60
61      # 0 -> amplitude A
62      # 1 -> amplitude B
63      # 2 -> amplitude C
64      # 3 -> distance
65      global observation
66      observation = np.zeros(4)
67
68      global mutex
69      mutex = QMutex()
70      global global_x
71      global_x = 0
72      global global_y
73      global_y = 0
74
75      # experimental values
76      global global_amplitude
77      global_amplitude = 0
78      global global_distance
79      global_distance = 0
80
81
82  def set_amplitude_a(num):
83      observation[0] = num
84
85
86  def set_amplitude_b(num):
87      observation[1] = num
88
89
90  def set_amplitude_c(num):
91      observation[2] = num
92
93
94  def set_distance(num):
```

```
 95     observation[3] = num
 96
 97
 98 def amplitude_a():
 99     return observation[0]
100
101
102 def amplitude_b():
103     return observation[1]
104
105
106 def amplitude_c():
107     return observation[2]
108
109
110 def distance():
111     return observation[3]
112
```

```
 1 import time
 2 from PyQt5.QtCore import QThread, QMutex
 3 from phasedetector import PhaseDetector
 4 from pykalman import KalmanFilter
 5 from math import sqrt, pow, sin, cos, pi, radians
 6 from enum import Enum
 7
 8 from scipy.stats import norm
 9 import numpy as np
10
11 # testing imports
12 from firebase import firebase
13
14 import globals
15
16
17 class Sector(Enum):
18     A = 0
19     B = 1
20     C = 2
21
22 # sector A - top left
23 amplitudeA = 10
24
25 # sector B - top right
26 amplitudeB = 0
27
28 # sector C - bottom
29 amplitudeC = 0
30
31
32 class DatabaseThread(QThread):
33     def __init__(self, email):
34         # only instantiated when an internet connection is
   detected
35         QThread.__init__(self)
36         self.email = email
37         # create JSON object to push to database
38         self.data = {
39             'amplitude': 0,
40             'distance': 0
41         }
42         self.setup = False
43         self.isOnline = True
44
45     def run(self):
46         while True:
47             if self.setup:
48                 globals.database.child("users")\
49                     .child("coordinates")\
```

```
50                      .update({"amplitude": globals.
   amplitudeA}, globals.user_token)
51                  globals.database.child("users")\
52                      .child("coordinates")\
53                      .update({"distance": globals.distance},
    globals.user_token)
54                  # print("Posting...")
55                  # self.data['amplitude'] = globals.
   global_amplitude
56                  # self.data['distance'] = globals.
   global_distance
57                  # globals.database.update(self.data,
   globals.user_token)
58              else:
59                  self.data['amplitude'] = globals.
   global_amplitude
60                  self.data['distance'] = globals.
   global_distance
61                  globals.database.child("users").child("
   coordinates").set(self.data, globals.user_token)
62                  # globals.database.push(self.data, globals.
   user_token)
63                  # globals.database.child("users").push(self
   .data, globals.user_token)
64                  # globals.database.child("coordinates").
   child("Lana").set(self.data, globals.user_token)
65                  self.setup = True
66              time.sleep(0.01)
67
68
69 class ADCThread(QThread):
70
71     def __init__(self):
72         QThread.__init__(self)
73         self.sample_size = 10
74         # addr = empty -> 0x48
75         # addr = sda -> 0x4a
76         # addr = scl -> 0x4b
77
78         # phase_detector0 setup
79         # A0 = amplitude
80         # A2 = distance
81         self.phase_detector0 = PhaseDetector(0x48)
82
83         # phase_detector1 setup
84         # A0 = amplitude
85         self.phase_detector1 = PhaseDetector(0x4a)
86
87         # additional phase detector
88         self.phase_detector2 = PhaseDetector(0x4b)
```

```
 89         self.kf = KalmanFilter(initial_state_mean=0,
    n_dim_obs=2)
 90
 91     def run(self):
 92         counter = 0
 93         while True:
 94             globals.mutex.lock()
 95
 96             # update previous globals
 97             globals.pAmplitudeA = globals.amplitudeA
 98             globals.pAmplitudeB = globals.amplitudeB
 99             globals.pAmplitudeC = globals.amplitudeC
100             #globals.prev_state_means = globals.
    curr_state_means
101             #globals.prev_covariances = globals.
    curr_covariances
102
103             amplitude_a = 0
104             amplitude_b = 0
105             amplitude_c = 0
106             distance = 0
107
108             # tolerable range: +/- 5 value
109
110             for i in range(0, self.sample_size):
111                 # set amplitude buffers using readings
    from phase detectors
112                 amplitude_a = amplitude_a + self.
    phase_detector0.read_channel_zero()
113                 # amplitude_b = amplitude_b + self.
    phase_detector0.read_channel_three()
114                 # amplitude_c = amplitude_c + self.
    phase_detector0.read_channel_zero()
115                 distance = distance + self.phase_detector0
    .read_channel_two()
116
117
118             #globals.prev_state_means = globals.
    curr_state_means
119             #globals.prev_covariances = globals.
    curr_covariances
120             #
121             # amplitude_a = self.phase_detector0.
    read_channel_zero()
122             # distance = self.phase_detector0.
    read_channel_two()
123
124             # from plot_online.py
125             # globals.curr_state_means, globals.
    curr_covariances = (
```

```
126             #    self.kf.filter_update(
127             #        globals.prev_state_means,
128             #        globals.prev_covariances#,
129             #        globals.observation
130             #    )
131             #)
132 
133             # TODO: Remove -1, only used for testing in sector A
134             globals.amplitudeA = amplitude_a / self.sample_size
135             globals.amplitudeB = amplitude_b / self.sample_size
136             globals.amplitudeC = amplitude_c / self.sample_size
137             globals.distance = distance / self.sample_size
138 
139             globals.mutex.unlock()
140             time.sleep(0.001)
141 
142 
143 class FilterThread(QThread):
144     @staticmethod
145     def polar_to_cartesian(rho, phi):
146         x = rho * cos(phi)
147         y = rho * sin(phi)
148         return x, y
149 
150     def quadratic(self, y):
151         root = pow(self.b, 2) + (4 * self.a * (y - self.c))
152         num = -self.b + root
153         den = 2 * self.a
154 
155         if num > 0:
156             num = -self.b - root
157             return num/den
158         else:
159             return num/den
160         # root = ((y-self.c)/self.a) + (pow(self.b, 2)/(4*pow(self.a, 2)))
161         # result = -(self.b/(2 * self.a)) + sqrt(root)
162         # return result
163 
164     def distance_quadratic(self, y):
165         root = pow(self.b_distance, 2) + (4 * self.a_distance * (y - self.c_distance))
166         num = -self.b_distance + root
167         den = 2 * self.a_distance
168 
```

```
169         if num < 0:
170             num = -self.b_distance - root
171             return num/den
172         else:
173             return num/den
174         # root = ((y-self.c)/self.a) + (pow(self.b, 2)/(4*
    pow(self.a, 2)))
175         # result = -(self.b/(2 * self.a)) + sqrt(root)
176         # return result
177 
178     def get_angle(self, x):
179         # raw angle
180         return_value = (self.a_angle * pow(x, 2)) + (self.
    b_angle * x) + self.c_angle
181         # factor = (max_plot_angle/
    maximum_observable_reading)
182         return return_value * (120/1400)
183 
184     def get_distance(self, x):
185         # raw distance
186         return_value = (self.a_distance * pow(x, 2)) + (
    self.b_distance * x) + self.c_distance
187         # factor = (max_plot_distance/
    maximum_observable_reading)
188         return return_value * (180/470)
189 
190     def update_globals(self, amplitude_reading, rho,
    reference_angle):
191         # Phi Calculation
192         voltage = (amplitude_reading * self.max_voltage) /
     self.resolution
193         phi = reference_angle - self.quadratic(voltage)
194         # phi = reference_angle - self.get_angle(voltage)
195 
196         # Rho Calculation
197         # phi = reference_angle - self.get_angle(voltage)
198         # max rho = 180
199         # max distance = 120 cm
200         distance_voltage = (rho * self.max_voltage) / self
    .resolution
201         rho_modified = self.get_distance(distance_voltage)
202 
203         coordinates = self.polar_to_cartesian(rho_modified
    , phi)
204 
205         if self.min_valid_voltage < voltage < self.
    max_valid_voltage and .45 < distance_voltage < 1.48:
206             # valid point
207             globals.global_x = coordinates[0]
208             globals.global_y = coordinates[1]
```

```
209         else:
210             # invalid point
211             globals.global_x = 0
212             globals.global_y = 220
213 
214     def sectorA(self):
215         # distance can be a max of 180, defined by plot
    size
216         # globals.distance = 90
217         # self.update_globals(globals.amplitudeA, globals.
    distance, 210)
218 
219         self.update_globals(globals.amplitudeA, globals.
    distance, 182)
220         # self.update_globals(globals.amplitudeA, globals.
    distance, 90)
221 
222     def sectorB(self):
223         self.update_globals(globals.amplitudeB, globals.
    distance, 90)
224 
225     def sectorC(self):
226         self.update_globals(globals.amplitudeC, globals.
    distance, -30)
227 
228     def __init__(self):
229         QThread.__init__(self)
230 
231         # uses quadratic function
232         # angle coefficients
233         self.a = -0.00001729241
234         self.b = -0.00437652647
235         self.c = 1.21484747253
236 
237         # new angle coefficients
238         self.a_angle = -162.73642899330
239         self.b_angle = 124.85705157978
240         self.c_angle = 84.50020263516
241 
242         # TODO: Save values
243         #self.a_angle = .23616
244         #self.b_angle = -179.0777
245         #self.c_angle = 218.48212
246 
247         # distance coefficients
248         self.a_distance = 503.50981468720
249         self.b_distance = -513.30730438075
250         self.c_distance = 132.01951515620
251 
252         self.max_voltage = 2.048
```

```
253         self.resolution = 2048
254         self.max_valid_voltage = 1.38
255         self.min_valid_voltage = 0.38
256         self.options = {
257             Sector.A: self.sectorA,
258             Sector.B: self.sectorB,
259             Sector.C: self.sectorC
260         }
261 
262     def get_sector(self):
263         if globals.amplitudeA > globals.amplitudeB:
264             if globals.amplitudeA > globals.amplitudeC:
265                 return Sector.A
266             else:
267                 return Sector.C
268         else:
269             if globals.amplitudeB > globals.amplitudeC:
270                 return Sector.B
271             else:
272                 return Sector.C
273 
274     def run(self):
275         while True:
276             sector = self.get_sector()
277             self.options[sector]()
278             # mutex.lock()
279             time.sleep(0.001)
280             # mutex.unlock()
281 
```

```python
from PyQt5.QtCore import QRectF, QSize, QPoint
from PyQt5.QtGui import QColor, QPainter, QPalette, QPen
from PyQt5.QtWidgets import QSizePolicy, QWidget
import globals


class Plot(QWidget):
    def __init__(self, parent=None):
        super(Plot, self).__init__(parent)

        self.floatBased = False
        self.antialiased = False
        self.frameNo = 0

        self.setBackgroundRole(QPalette.Base)
        self.setSizePolicy(QSizePolicy.Expanding,
QSizePolicy.Expanding)

        self.alpha = 0
        self.radius = 30
        self.rings_plotted = False
        self.setup_finished = False
        self.x = 0
        self.y = 0

        self.paintEvent = self.draw_rings
        self.tracking = True

        self.line_color = '#4080fe'
        self.point_color = '#db2c38'

    def setFloatBased(self, floatBased):
        self.floatBased = floatBased
        self.update()

    def setAntialiased(self, antialiased):
        self.antialiased = antialiased
        self.update()

    def minimumSizeHint(self):
        return QSize(50, 50)

    def sizeHint(self):
        return QSize(180, 180)

    def point(self, x, y):
        self.x = x
        self.y = y

    def nextAnimationFrame(self):
```

```
50          if self.setup_finished:
51              if self.tracking:
52                  self.x = globals.global_x
53                  self.y = globals.global_y
54              self.update()
55          else:
56              if self.rings_plotted:
57                  # read data and plot
58                  if self.alpha >= 240:
59                      self.alpha = 255
60                      self.update()
61                      self.paintEvent = self.draw_point
62                      self.setup_finished = True
63                  else:
64                      self.alpha += 20
65                      self.update()
66              else:
67                  if self.radius > 390:
68                      self.paintEvent = self.draw_lines
69                      self.rings_plotted = True
70                      self.alpha = 0
71                      self.update()
72                  else:
73                      self.alpha = (self.alpha + 20) % 260
74                      if self.alpha == 0:
75                          self.radius += 30
76                      self.update()
77
78      def change_colors(self, line, point):
79          self.line_color = line
80          self.point_color = point
81
82      def continue_tracking(self):
83          self.tracking = True
84
85      def pause_tracking(self):
86          self.tracking = False
87
88      def refresh(self):
89          self.floatBased = False
90          self.antialiased = False
91          self.frameNo = 0
92
93          self.alpha = 0
94          self.radius = 30
95          self.rings_plotted = False
96          self.setup_finished = False
97          self.x = 0
98          self.y = 0
99
```

```
100         self.paintEvent = self.draw_rings
101         self.tracking = True
102 
103     def setup_plot(self, event):
104         color = QColor(0, 0, 0)
105         # color.setNamedColor('#4080fe')
106         color.setNamedColor(self.line_color)
107         color.setAlpha(self.alpha)
108 
109         painter = QPainter(self)
110         painter.setPen(color)
111         painter.setRenderHint(QPainter.Antialiasing, self.
    antialiased)
112         painter.translate(self.width() / 2, self.height()
    / 2)
113 
114         for diameter in range(0, 390, 30):
115             delta = abs((40 % 128) - diameter / 2)
116             alpha = 255 - (delta * delta) / 4 - diameter
117             painter.drawEllipse(QRectF(-diameter / 2.0, -
    diameter / 2.0, diameter, diameter))
118 
119         for l in range(0,180,1):
120             painter.drawPoint(0,-l)
121 
122         i = 0
123         step_x = .8660254
124         step_y = .5
125 
126         # 180 is a fixed bound
127         for i in range(0,180,1):
128             painter.drawPoint(i * step_x, i * step_y)
129             painter.drawPoint(-i * step_x, i * step_y)
130 
131     # draw rings new
132     def draw_rings(self, event):
133         color_solid = QColor(0, 0, 0)
134         # color_solid.setNamedColor('#4080fe')
135         color_solid.setNamedColor(self.line_color)
136 
137         color_light = QColor(0, 0, 0)
138         color_light.setNamedColor(self.line_color)
139         # color_light.setNamedColor('#4080fe')
140         color_light.setAlpha(self.alpha)
141 
142         painter = QPainter(self)
143         painter.setPen(color_light)
144         painter.setRenderHint(QPainter.Antialiasing, self.
    antialiased)
145         painter.translate(self.width() / 2, self.height()
```

```
145 / 2)
146 
147         # max should be relative, fixed at 390 right now
148         # for diameter in range(0, self.radius, 30):
149         for diameter in range(0, 390, 30):
150 
151             if diameter <= self.radius:
152                 # draw whole
153                 painter.setPen(color_solid)
154 
155                 delta = abs((40 % 128) - diameter / 2)
156                 alpha = 255 - (delta * delta) / 4 -
    diameter
157                 painter.drawEllipse(QRectF(-diameter / 2.0
    , -diameter / 2.0, diameter, diameter))
158             else:
159                 painter.setPen(color_light)
160 
161                 delta = abs((40 % 128) - diameter / 2)
162                 alpha = 255 - (delta * delta) / 4 -
    diameter
163                 painter.drawEllipse(QRectF(-diameter / 2.0
    , -diameter / 2.0, diameter, diameter))
164                 break
165 
166     def draw_lines(self, event):
167 
168         color = QColor(0, 0, 0)
169         # color.setNamedColor('#4080fe')
170         color.setNamedColor(self.line_color)
171 
172         painter = QPainter(self)
173         painter.setPen(color)
174         painter.setRenderHint(QPainter.Antialiasing, self.
    antialiased)
175         painter.translate(self.width() / 2, self.height()
    / 2)
176 
177         for diameter in range(0, 390, 30):
178             delta = abs((40 % 128) - diameter / 2)
179             alpha = 255 - (delta * delta) / 4 - diameter
180             painter.drawEllipse(QRectF(-diameter / 2.0, -
    diameter / 2.0, diameter, diameter))
181 
182         color.setAlpha(self.alpha)
183         painter_light = QPainter(self)
184         painter_light.setPen(color)
185         painter_light.setRenderHint(QPainter.Antialiasing,
     self.antialiased)
186         painter_light.translate(self.width() / 2, self.
```

```
height() / 2)

        for l in range(0,180,1):
            painter_light.drawPoint(0,-l)

        i = 0
        step_x = .8660254
        step_y = .5

        # 180 is a fixed bound
        for i in range(0,180,1):
            painter_light.drawPoint(i * step_x, i * step_y
)
            painter_light.drawPoint(-i * step_x, i *
step_y)

    def draw_point(self, event):
        color = QColor(0, 0, 0)
        # color.setNamedColor('#4080fe')
        color.setNamedColor(self.line_color)

        painter = QPainter(self)
        painter.setPen(color)
        painter.setRenderHint(QPainter.Antialiasing, self.
antialiased)
        painter.translate(self.width() / 2, self.height()
/ 2)

        for diameter in range(0, 390, 30):
            delta = abs((40 % 128) - diameter / 2)
            alpha = 255 - (delta * delta) / 4 - diameter
            painter.drawEllipse(QRectF(-diameter / 2.0, -
diameter / 2.0, diameter, diameter))

        # draw lines
        for l in range(0,180,1):
            painter.drawPoint(0,-l)

        step_x = .8660254
        step_y = .5

        # add degree labels
        font = painter.font()
        font.setPointSize(10)
        painter.setFont(font)

        # degree labels
        painter.drawText(QPoint(5, -167), "120\xb0")
        painter.drawText(QPoint(-15, -167), "0\xb0")
        painter.drawText(QPoint(-175, 85), "120\xb0")
```

```
231         painter.drawText(QPoint(-157, 105), "0\xb0")
232         painter.drawText(QPoint(150, 85), "0\xb0")
233         painter.drawText(QPoint(138, 105), "120\xb0")
234 
235         font.setPointSize(8)
236         painter.setFont(font)
237 
238         # distance labels
239         painter.drawText(QPoint(-7, 25), "0.5")
240         painter.drawText(QPoint(-7, 40), "1.0")
241         painter.drawText(QPoint(-7, 55), "1.5")
242         painter.drawText(QPoint(-7, 70), "2.0")
243         painter.drawText(QPoint(-7, 85), "2.5")
244         painter.drawText(QPoint(-7, 100), "3.0")
245         painter.drawText(QPoint(-7, 115), "3.5")
246         painter.drawText(QPoint(-7, 130), "4.0")
247         painter.drawText(QPoint(-7, 145), "4.5")
248         painter.drawText(QPoint(-7, 160), "5.0")
249         painter.drawText(QPoint(-7, 175), "5.5")
250 
251 
252         # 180 is a fixed bound
253         for i in range(0, 180, 1):
254             painter.drawPoint(i * step_x, i * step_y)
255             painter.drawPoint(-i * step_x, i * step_y)
256 
257         # color.setNamedColor('#db2c38')
258         color.setNamedColor(self.point_color)
259         painter_red = QPainter(self)
260         painter_red.setPen(color)
261         painter_red.setRenderHint(QPainter.Antialiasing,
    self.antialiased)
262         painter_red.translate(self.width()/2, self.height(
    )/2)
263 
264         for i in range(0, 6, 1):
265             painter_red.drawPoint(self.x+i, self.y)
266             painter_red.drawPoint(self.x+i, self.y+1)
267             painter_red.drawPoint(self.x+i, self.y-1)
268 
269             painter_red.drawPoint(self.x-i, self.y)
270             painter_red.drawPoint(self.x-i, self.y+1)
271             painter_red.drawPoint(self.x-i, self.y-1)
272 
273             painter_red.drawPoint(self.x, self.y+i)
274             painter_red.drawPoint(self.x+1, self.y+i)
275             painter_red.drawPoint(self.x-1, self.y+i)
276 
277             painter_red.drawPoint(self.x, self.y-i)
278             painter_red.drawPoint(self.x+1, self.y-i)
```

```
279                 painter_red.drawPoint(self.x-1, self.y-i)
280
```

```
# File: adc.py
#
# Name: Carlos Rivera
# Description:
#   The PhaseDetector class allows for easy reading of ADC output
#   and allows the phase and amplitude readings to be easily
#   distinguished and read. Each ADC should be setup in the
#   way described inside application.py. At the bare minimum,
#   the first address should correspond to the amplitude, the
#   second to the phase, and finally the last to the remaining
#   analog output from the additional phase detector.

import Adafruit_ADS1x15


class PhaseDetector:

    def __init__(self, addressin=0x48, busnumin=1):
        self.adc = Adafruit_ADS1x15.ADS1015(address=addressin, busnum=busnumin)
        self.GAIN = 2

    # read amplitude
    def read_channel_zero(self):
        return self.adc.read_adc(0, gain=self.GAIN)

    # read phase
    def read_channel_one(self):
        return self.adc.read_adc(1, gain=self.GAIN)

    # read amplitude or phase of remaining antenna
    def read_channel_two(self):
        return self.adc.read_adc(2, gain=self.GAIN)

    def read_channel_three(self):
        return self.adc.read_adc(3, gain=self.GAIN)
```

I Senior Design Conference Slides as Presented

RF Location Tracking

A Modular Antenna System Implementation

Alan Tao Feng, Carlos Rivera, Razma Mogharrab

Advisors: R. Abhari Ph.D, A. Amer Ph.D

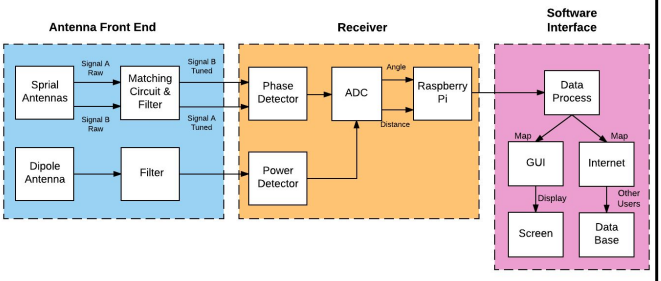
Problem Statement

- **Current RF drone positioning systems:**
  - GPS:
    - High error margins (4 meter resolution at best)
    - Does not work indoors
    - Requires probability-based, error-correcting code
  - Wi-fi:
    - Requires multiple wireless access points
    - Large-scale, extensive environmental setup
- **Proposed Solution**
  - Use a single RF receiver system
  - Track the telemetry signal of the drone

Objectives

- **Create a modular RF location tracking system**
  - Detect an incoming RF source
  - Locate the RF source (relative distance and direction)
  - Focus on 915 MHz frequency due to use in unmanned aerial vehicles (UAVs)
- **Display location via a Raspberry Pi 3**
  - Positioning displayed on a graphical user interface
  - Display information with a reasonable refresh rate

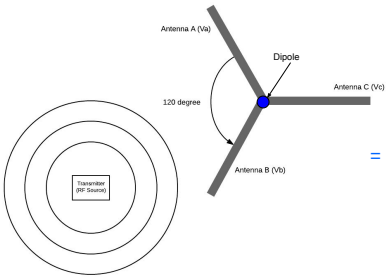
Functional Block Diagram



Requirements

Functional	Nonfunctional
= Operate in the ultra high frequency (UHF - 915MHz) range	= Be a modular system
= Detect RF sources for distances up to 6 meters with a resolution of 5 centimeters	= Have a scalable nature
= Have a maximum angular error of 1 degree (10 centimeters arc length at max distance) in 120° sector	= Have core functionality be internet independent
	= Be low cost and use off-the-shelf components

Location Tracking Method

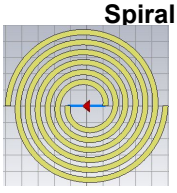


= **Direction**
- Three separate bi-directional spiral antennas are placed 120° apart
- RF source angle determined by comparing the difference in received power between the antennas

= **Distance**
- Uses an omni-directional dipole antenna
- Distance determined by received power

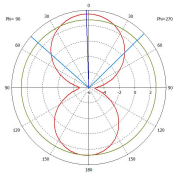
Spiral Antenna

Spiral antenna in CST

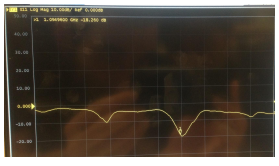


Actual antenna used

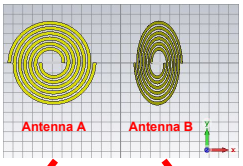
Radiation pattern

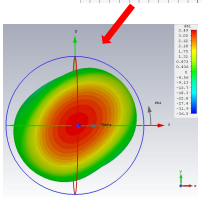


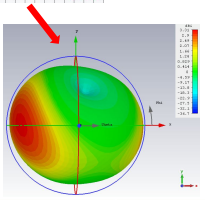
S - Parameter (VNA)



Determine Direction







= **As RF source move away from the center**
- Antenna A received power decreases
- Antenna B received power increase

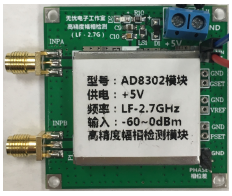
= **Use the ratio between the received power levels to determine angle of RF source**

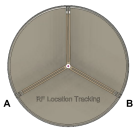
Phase Detector

- Using Analog Devices AD8302 Phase Detector
- Receives power from antennas A and B
- Gain measurement scaling: 30mV/dB

Antenna A

Antenna B





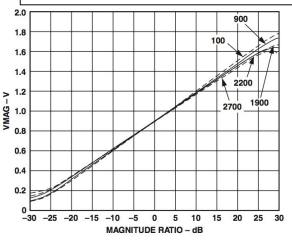
$$V_{mag} = 20log(\frac{V_{IN,A}}{V_{IN,B}})$$

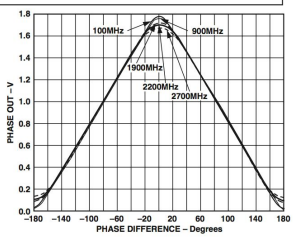
Phase Detector Module

ANALOG DEVICES

LF–2.7 GHz
RF/IF Gain and Phase Detector

AD8302

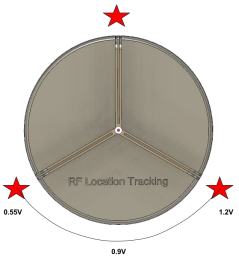




Receiver Module - Numbers Test

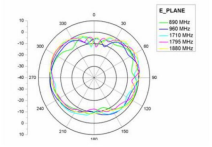
Amplitude Ratio	DC Output (Theoretical)	DC Output (Experimental)
20log(Va/Vb) = -30dB	30mV	30.2mV
Va = Vb	900mV	904mV
20log(Va/Vb) = +30dB	1.8V	1.805V
Phase Difference	**DC Output (Theoretical)**	**DC Output (Experimental)**
0°	30mV	30.1mV
±90°	900mV	860mV
±180°	1.8V	1.779V

Sector Determination

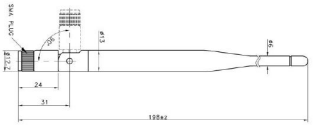


- Use one phase detector per sector
- The phase detector outputs a voltage between 0.55V and 1.2V determines the sector that the RF source is in
- The antennas that connect to that phase detector are the bounds of the sector in which the RF source lies

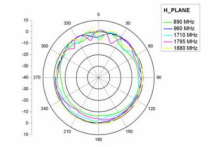
Dipole Antenna



Quad-band Cellular Antenna



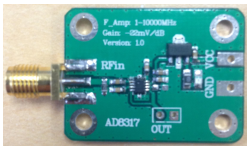
Radiation pattern

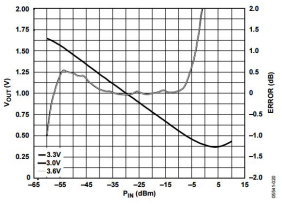


S - Parameter (VNA)

Determine Distance

- Using Analog Devices AD8317 chip to determine received power as a function of separation distance

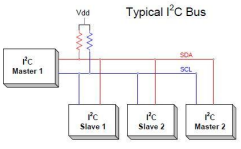




Raspberry Pi Interfacing

- Phase detector output passed to ADS1015 12-bit ADC with programmed output voltage range
- Communication via I^2C
- Interfacing with ADC done with Adafruit Python ADS1x15 library

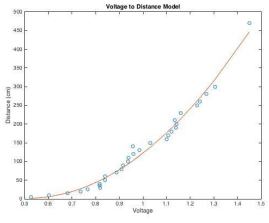
Typical I^2C Bus

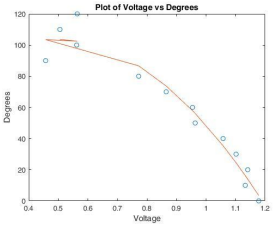


Raspberry Pi Application

- Written in Python
- User interface made using PyQt5
- Relative layout allows for common display size compatibility
- Pyrebase library used to interact with database

Data Collection

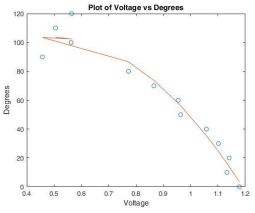


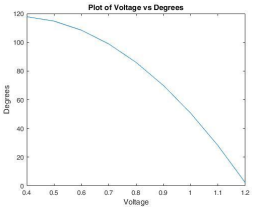


Distance = 503.509x^2 - 513.307x + 132.019
= (503.509x - 513.307)x + 132.019

Angle = -209.936x^2 + 204.645x + 53.808
= (-209.936x + 204.645)x + 53.808

Final Angle Equation

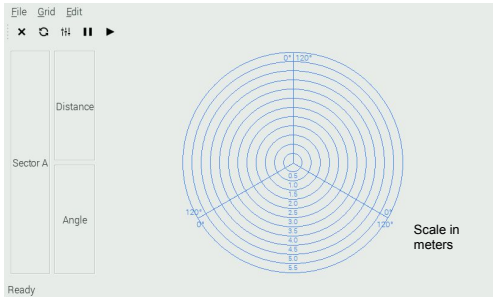




Angle = -209.936x^2 + 204.645x + 53.808
= (-209.936x + 204.645)x + 53.808

Angle = -162.7364x^2 + 115.857x + 97.500
= (-162.7364x + 115.857)x + 97.500

Graphical User Interface



Realtime Database

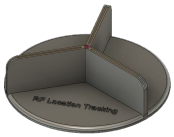
- Database stores raw ADC output values
- Unique data for each registered user
- Allows for data use on additional devices
- An extension for project goal

Firebase

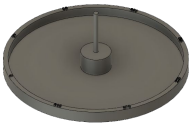
Testing Base

- Designed and 3D printed holder and rotary axis for antenna array

Y-Holder Model 1



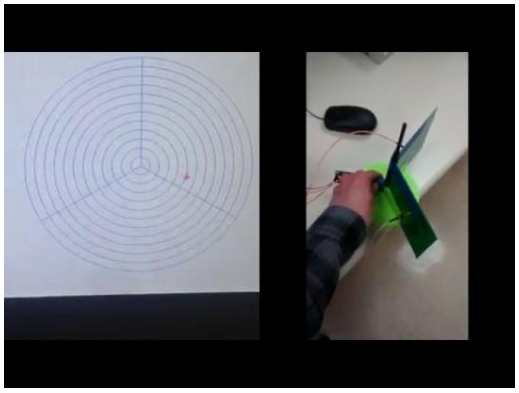
Y-Holder Model 2



Y-Spinner

System Testing

Scale: Rings are 3 cm apart



Using Portable Transmitter

- System is too sensitive to change in received power
  - Different transmitting power causes inconsistencies
  - Does not track our portable transmitter as well because it does not transmit at constant power level at all times
  - Movement in the environment causes inconsistencies
- Solutions
  - Better Filtering
    - Kalman Filter (Software)
    - Antenna Front End (Hardware)
  - Received Signal Strength Indicator (RSSI)/Demodulator

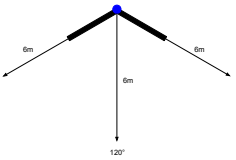
Portable RF Transmitter

Future Work

- Assemble a three module unit
- Compartmentalize and minimize receiver/transmitter modules
- Have system support 3D movement/elevation
- Test drone mounting/compatibility

Accomplishments

- Created a modular RF location tracking system
- Tracks within 1 sector
  - Determine if an RF source is in that particular sector
  - Track the location of RF source in that sector
- Specs
  - Full area coverage of 38 m^2 (120°)
  - Maximum angular error of 1°
  - Minimum distance resolution of 5 centimeters



Special Thanks to:

Shivam Gandhi

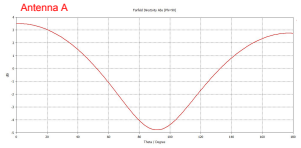
Thank you.

Questions?

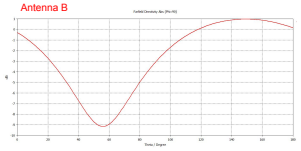
Conflict Resolution (Angle)

Gain vs Angle (Cartesian)

Antenna A



Antenna B



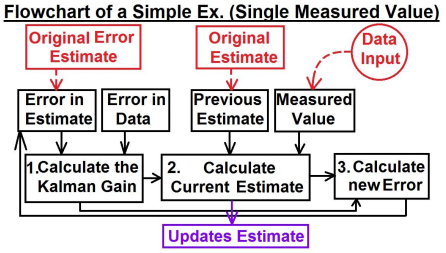
Based on

- Change in gain as a function of angle
- Sensitivity of RF Gain Phase Detector
  - Minimum detectable power level difference
  - Minimum difference in output voltage

Iterative Kalman Filtering

- Kalman Gain used in determining validity of new observations with respect to previous.

Flowchart of a Simple Ex. (Single Measured Value)

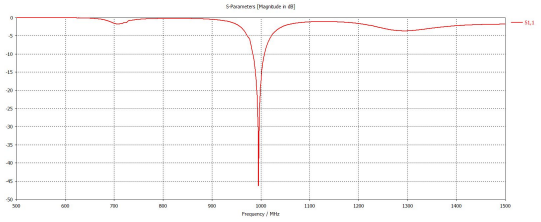


Useful Numbers/Information

- GPS error - 3m to 4m, requires probability code to lessen
- Drone operating frequency: 433MHz (Europe), 915MHz (USA), 2.4GHz (Wi-fi communication) and 5.8GHz are common too.
- Current tracking systems: Wi-fi zones (setup large perimeter and triangulate), GPS, or cameras (image processing)
- Sensitivity of phase detector:

Useful Numbers/Information

- Far Field for Spiral Antenna = 16cm
- Far Field for Dipole Antenna = 32cm



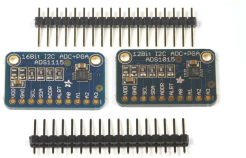
ADC Comparison

ADS 1015 Features

- Resolution: 12 Bits
- Programmable Sample Rate: 128 to 3300 Samples/Second

ADS 1115 Features

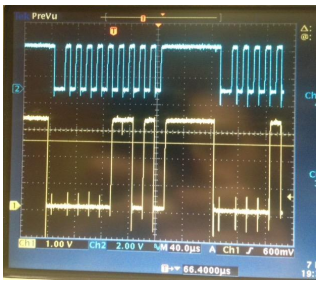
- Resolution: 16 Bits
- Programmable Sample Rate: 8 to 860 Samples/Second



ADC Component Testing

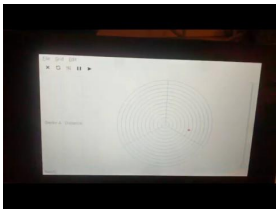
ADC Clock:

ADC Data Line:

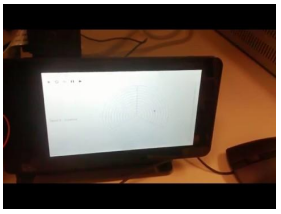


Stationary RF Source Comparison

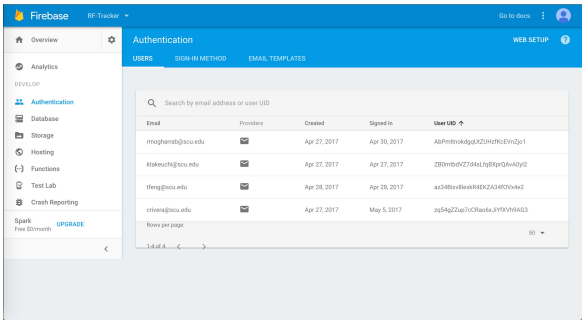
With Function Generator



With Portable Transmitter



Firebase Authenticated User List



Values Saved Per User